The Ammasso 1100

High Performance Ethernet Adapter

User's Guide

Ammasso, Inc.
345 Summer St.
Boston, MA 02210
Main: 617-532-8100
Fax: 617-532-8199

http://www.ammasso.com
Revision: 1.2Update 1
July 22, 2005

Copyright © 2005 Ammasso, Inc. All rights reserved.

Information in this document is subject to change without notice.

This document is provided for information only. Ammasso, Inc. makes no warranties of any kind regarding the Ammasso 1100 High Performance Ethernet Adapter except as set forth in the license and warranty agreements. The Ammasso 1100 High Performance Ethernet Adapter is the exclusive property of Ammasso, Inc. and is protected by United States and International copyright laws. Use of the product is subject to the terms and conditions set out in the accompanying license agreements. Installing the product signifies your agreement to the terms of the license agreements.

Table of Contents

1 OVERVIEW 6

1.1 INTRODUCTION 6
1.2 THEORY OF OPERATION 7
1.2.1 HOW THE AMMASSO ADAPTER WORKS 7
1.2.2 HOW REMOTE DIRECT MEMORY ACCESS (RDMA) WORKS 7
1.2.3 WHAT IS MPI AND WHY IS IT USED? 9
1.2.4 WHAT IS DAPL AND WHY IS IT USED? 9
1.3 PRODUCT COMPONENTS 10
1.4 SPECIFICATIONS 10
1.4.1 PERFORMANCE 10
1.4.2 APPLICATION PROGRAM INTERFACES (APIS) 10
1.4.3 OPERATING SYSTEMS 10
1.4.4 PLATFORMS 10
1.4.5 MANAGEMENT 10
1.4.6 STANDARDS COMPLIANCE 10
1.4.7 HARDWARE 10

2 HARDWARE SYSTEM REQUIREMENTS AND INSTALLATION 11

2.1 SAFETY AND EMISSIONS 11
2.2 SYSTEM HARDWARE REQUIREMENTS 12
2.3 ADAPTER HARDWARE INSTALLATION 12
2.3.1 CHOOSING A SLOT FOR INSTALLATION 12
2.3.2 TOOLS AND EQUIPMENT REQUIRED 12
2.3.3 ADAPTER INSERTION PROCEDURE 13

3 ADAPTER SOFTWARE INSTALLATION 14

3.1 SYSTEM SOFTWARE REQUIREMENTS 14
3.2 OVERVIEW 14
3.3 INSTALLING THE ADAPTER SOFTWARE PACKAGE 14
3.3.1 MAKEFILE TARGETS 16
3.3.2 MAKEFILE CONFIGURATION VARIABLES 17
3.4 DISTRIBUTION SPECIFIC BUILD DETAILS 18
3.4.1 REDHAT 18
3.4.2 SUSE 19
3.4.3 KBUILD 20
AMMASSO 1100 COMMANDS AND UTILITIES 21
3.5 CONFIGURING THE AMMASSO 1100 ADAPTER 22
3.5.1 CONFIGURATION ENTRIES 23
3.5.2 SAMPLE CONFIGURATION FILE 24
3.6 VERIFYING THE ADAPTER SOFTWARE INSTALLATION 24
3.7 REMOVING AN ADAPTER SOFTWARE INSTALLATION 25

4 THE AMMASSO MPI LIBRARY 26

4.1 OVERVIEW 26
4.1.1 COMPILER SUPPORT 26
4.2 INSTALLATION 26
4.2.1 MAKEFILE TARGETS 28
4.2.2 MAKEFILE CONFIGURATION VARIABLES 29
4.3 LOCATING THE MPI LIBRARIES AND FILES 30
4.4 COMPILING AND LINKING APPLICATIONS 31
4.4.1 CREATING A MPI CLUSTER MACHINES FILE 32
4.4.2 REMOTE STARTUP SERVICE 33
4.5 VERIFYING MPI INSTALLATION 33
4.6 REMOVING THE AMMASSO MPI INSTALLATION 34
4.7 AMMASSO MPI TUNABLE PARAMETERS 34
4.7.1 VIADEV ENVIRONMENT VARIABLES 35
4.7.2 TUNING SUGGESTIONS 37

5 THE AMMASSO DAPL LIBRARY 38

5.1 OVERVIEW 38
5.2 INSTALLATION 38
5.2.1 MAKEFILE TARGETS 39
5.2.2 MAKEFILE CONFIGURATION VARIABLES 39
5.3 CONFIGURING DAPL 40
5.4 VERIFYING DAPL INSTALLATION 40
5.4.1 UDAPL INSTALLATION VERIFICATION 40
5.4.2 KDAPL INSTALLATION VERIFICATION 41
5.5 REMOVING THE AMMASSO DAPL INSTALLATION 41
5.6 AMMASSO DAPL COMPATIBILITY SETTINGS 41
5.6.1 CCAPI_ENABLE_LOCAL_READ 41
5.6.2 DAPL_1_1_RDMA_IOV_DEFAULTS 42

6 CLUSTER INSTALLATION 43

6.1 INTRODUCTION 43
6.2 STEPS ON THE INITIAL BUILD SYSTEM 43
6.2.1 PREPARE THE KERNEL 43
6.2.2 INSTALL AMSO1100 AND BUILD BINARY FOR CLUSTER DEPLOYMENT 44
6.2.3 INSTALL MPICH AND BUILD BINARY FOR CLUSTER DEPLOYMENT 46
6.2.4 INSTALL DAPL AND BUILD BINARY FOR CLUSTER DEPLOYMENT 48
6.2.5 CLEAN UP AMSO DIRECTORIES AND FILES. 49
6.3 STEPS ON THE CLUSTER NODE SYSTEMS 49
6.4 CLUSTER DEPLOYMENT 52

7 USING THE AMMASSO 1100 WITH PXE BOOT 53

7.1 THEORY OF OPERATION 53
7.2 REQUIREMENTS 53
7.3 BIOS SETTINGS 55
7.4 BUILDING THE RAMDISK 55
7.4.1 CONFIGURING AND BUILDING BUSYBOX APPLICATIONS 56
7.4.2 BUILDING MODUTILS OR MODULE-INIT-TOOLS 56

7.4.3 POPULATING THE RAMDISK 57
7.4.4 BUILDING THE RAMDISK IMAGE 59
7.5 INSTALLING AND CONFIGURING A DHCP SERVER 59
7.6 INSTALLING AND CONFIGURING A TFTP SERVER 60
7.7 BUILDING PXELINUX 61
7.8 DISKLESS LINUX BOOT VIA PXE 61
7.8.1 CONFIGURE A ROOT FILE SYSTEM 62
7.8.2 CONFIGURE THE NFS SERVER 62
7.9 UPDATING THE AMMASSO 1100 OPTION ROM IMAGE 63

APPENDIX A: SUPPORT 64

OBTAINING ADDITIONAL INFORMATION 64
CONTACTING AMMASSO CUSTOMER SUPPORT 64
RETURNING A PRODUCT TO AMMASSO 64

APPENDIX B: WARRANTY 65

1 Overview

1.1 Introduction

The Ammasso 1100 High Performance Ethernet Adapter is a RDMA enabled Ethernet Server adapter offering high performance for server to server networking environments. The Ammasso 1100 is also a gigabit Ethernet adapter and works within any gigabit Ethernet environment supporting existing standard wiring, switches, and other Ethernet adapters.

Using the Ammasso adapter requires installing it into a PCI-X slot, loading the software and assigning IP addresses to the adapter, one for RDMA traffic and one for standard sockets traffic. The adapter has several features that deliver higher levels of performance:

- Low latency
- Reduced copies
- Reduced host CPU overhead

These features are accomplished through the use of RDMA (Remote Direct Memory Access) technology and CPU offload. RDMA moves data from the memory of one server directly into the memory of another server over a network without involving the CPU of either server. CPU offload lowers the processing requirements on the host CPU when using RDMA, increasing its efficiency. Low latency assures that the time it takes for data to get from one application to the other is minimized, while increasing the overall message capacity of the network.

The combination of these features reduces application data transfer times. They combine to enable distributed databases to scale more effectively improving performance on individual servers and providing better scalability to larger clusters. Compute clusters attain higher performance levels with improved inter-node communication and reduced CPU loads. Lastly for traditional network attached storage (NAS) installations, storage solutions deliver data faster without additional CPU overhead..

1.2 Theory of Operation

1.2.1 How the Ammasso Adapter Works

The Ammasso 1100 is an Ethernet adapter designed to take advantage of various standard interfaces, including MPI, DAPL, and traditional BSD sockets.

For use with MPI and/or DAPL, the Ammasso 1100 leverages its RDMA capabilities and uses its on-board processing engine to quickly decipher the header information, determines where the information needs to go, and handles any network processing that needs to be done without involving the host CPU. Through this approach, the adapter limits the need to make data copies, limits the amount of host CPU processing necessary, and places data directly into application memory, thereby maximizing performance.

Additionally, the adapter can be used to support sockets based traffic as well. When using the BSD sockets interface, the Ammasso 1100 operates at performance levels consistent with other high-end gigabit sockets based adapters. The adapter supports sockets by maintaining a separate IP address within the card for sockets traffic and rapidly moving that traffic to the host network stack, where it can be processed normally.

Having support for both a high performance path and sockets path allows a single cable connection and switch data port for all traffic types, either sockets or fast-path RDMA based, simplifying network environments and management.

1.2.2 How Remote Direct Memory Access (RDMA) Works

Once a connection has been established, RDMA enables the movement of data from the memory of one server directly into the memory of another server without involving the operating system of either node. RDMA supports zero-copy networking by enabling the network adapter to transfer data directly to or from application memory, eliminating the need to copy data between application memory and the data buffers in the operating system. When an application performs an RDMA Read or Write request, the application data is delivered directly to the network, hence latency is reduced and applications can transfer messages faster (see Figure 1).

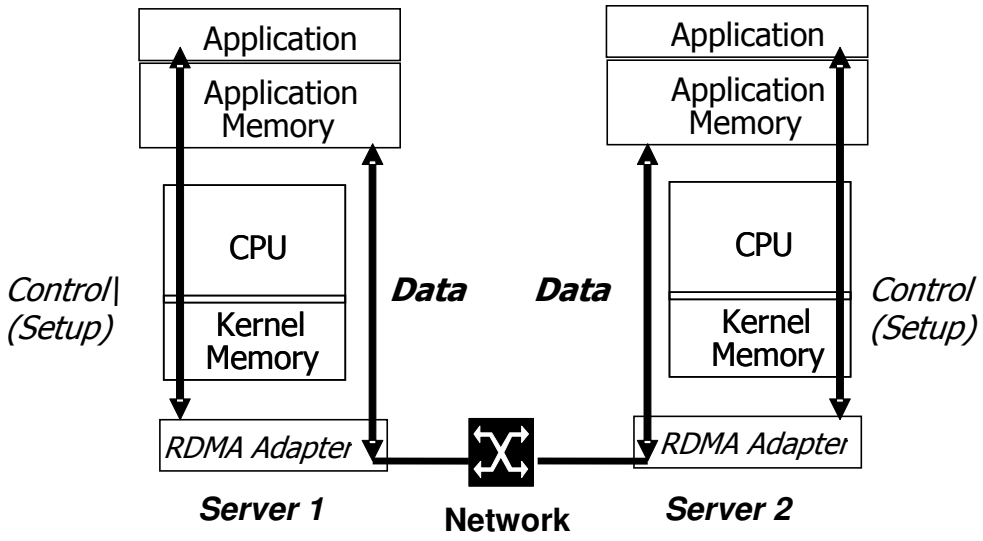

Figure 1 Transfer of Data

RDMA reduces demand on the host CPU by enabling applications to directly issue data transfer request commands to the adapter without having to execute an operating system call (referred to as "kernel bypass"). The RDMA request is issued from an application running on one server to the local adapter through its API and that request, along with the application's data, is then carried over the network to the remote adapter. The remote adapter then places the application's data into the host's memory without requiring operating system involvement at either end. Since all of the information pertaining to the remote virtual memory address is contained in the RDMA message itself, and host and remote memory protection issues were checked during connection establishment, the operating systems do not need to be involved in each message.

For each RDMA command, the Ammasso 1100 adapter implements all of the required RDMA operations as well as the processing of the TCP/IP state machine, thus reducing demand on the CPU and providing a significant advantage over standard adapters while maintaining TCP/IP standards and functionality. (see Figure 2).

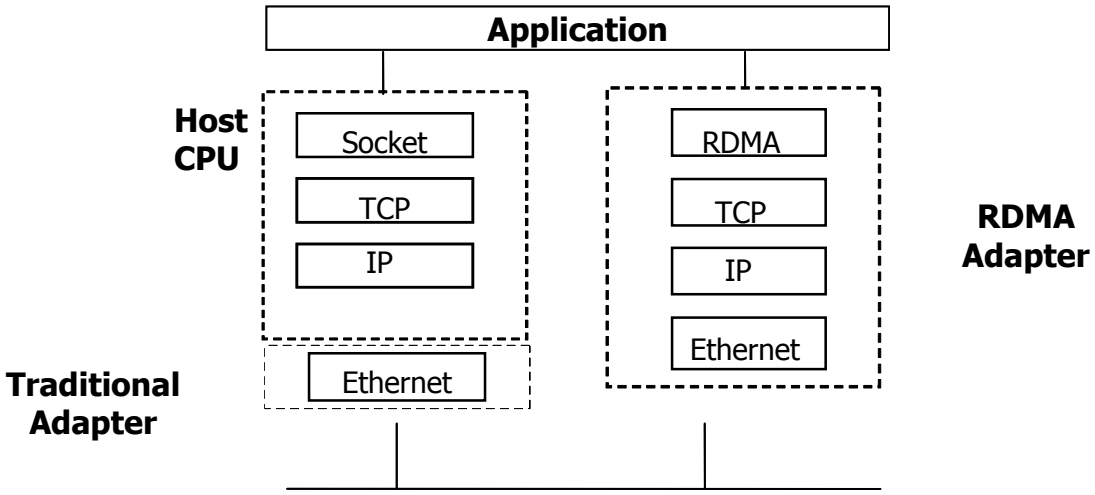

Figure 2 RDMA vs. Traditional Adapter

1.2.3 What Is MPI and Why Is It Used?

The Message Passing Interface (MPI) is a library for message-passing used for efficient distributed computation. Developed as a collaborative effort by a mix of research and commercial users, MPI has expanded into a standard supported by a broad set of vendors, implementers, and users and is designed for high performance applications on massively parallel machines and workstation clusters. It is widely available in a number of different vendor implementations, and open source distributions. MPI is considered the de-facto standard for writing parallel applications.

A version of the MPI library MPICH-1.2.5 is available that is compatible with the Ammasso 1100 adapter.

1.2.4 What Is DAPL and Why Is It Used?

Direct Access Programming Library (DAPL) defines a single set of user APIs used in Database and Storage RDMA Enabled environments. The DAPL API is defined by the DAT (Direct Access Transport) Collaborative which is an industry group that has formed in order to define and standardize a set of transport-independent, platform-independent Application Programming Interfaces (APIs) that exploit RDMA (remote direct memory access). It is being supported and utilized in NAS/SAN and Enterprise Database environments.

1.3 Product Components

- AMSO-1100 High Performance Ethernet Adapter
- A `.tgz` archive file containing all software files
- The Ammasso 1100 High Performance Ethernet Adapter User's Guide (this file)
- Current release notes

1.4 Specifications

1.4.1 Performance

- 1 Gigabit Ethernet
- Full duplex
- TCP/IP offload for RDMA Connections
- Full RDMAP Offload with MPA

1.4.2 Application Program Interfaces (APIs)

- MPI – Argonne National Labs MPICH Version 1.2.5
- DAPL – User Mode (uDapl) and Kernel Mode (kDapl) ver 1.2
- BSD Sockets (non-RDMA)

1.4.3 Operating Systems

- See `http://www.ammasso.com/support/`

1.4.4 Platforms

- IA-32 and EM64T compatible systems
- AMD Opteron 32/64 compatible systems

1.4.5 Management

- Field upgradeable
- PXE boot

1.4.6 Standards Compliance

- IEEE Ethernet: 802.3, 802.3ab (copper)
- PCI: PCI-X 1.0

1.4.7 Hardware

- Connector: Cat 5E cable terminated with an RJ45 connector
- Regulatory: FCC Class A, CE, CSA
- Size: Full height, PCI short card
- Temperature: 0° to 70°C (32° to 158°F)
- Humidity: 10 - 90% non-condensing

2 Hardware System Requirements and Installation

2.1 Safety and Emissions

FCC Class A Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interferences when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the User Guide, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

EN 55022 Class A Warning Requirements

Warning

This is a Class A Product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Observe the following precautions when working with your Ammasso 1100 High Performance Ethernet Adapter.

WARNING: Only trained and qualified personnel should be allowed to install or replace this equipment.

WARNING: Before performing the adapter insertion procedure, ensure that power is removed from the computer.

CAUTION: ***To prevent electrostatic damage (ESD), handle the adapter by its edges, and always use an ESD-preventative wrist strap or other grounding device.***

2.2 System Hardware Requirements

- Intel IA-32 and EM64T or AMD Opteron 32/64 compatible
- Bus Type: PCI-X 1.0
- Bus Width: 64-bit
- Bus Slot Speed: 66/100/133 MHz
- Power supply voltages:+3.3 volts +/- 0.3 volts, +5.0 volts +/- 0.25 volts
- A single open 64-bit PCI-X slot (see section "Choosing a Slot for Maximizing Performance" below before making a choice)
- 256 MB RAM (minimum), 1GB (or more recommended)
- PCI-X compatible riser card (if installing in a system that requires a riser card)

2.3 Adapter Hardware Installation

2.3.1 Choosing a Slot for Installation

When choosing a connector slot in your computer, be aware that the Ammasso 1100 Ethernet Adapter must be inserted into a PCI-X slot. Before inserting the adapter, use your system documentation to determine which slots support PCI-X. Be sure to set any jumpers correctly on the motherboard to ensure that the slot is correctly configured. The Ammasso 1100 will not operate correctly in a slot that is running in PCI mode. If a PCI device is attached to a PCI-X bus, the bus will operate in PCI mode.

The Ammasso 1100 Ethernet Adapter is designed to deliver high performance. In order to maximize performance, Ammasso recommends that the 1100 Ethernet Adapter be the only device on the PCI-X bus. Installing more than one card in a PCI-X bus may degrade performance of all devices on that bus.

2.3.2 Tools and Equipment Required

You need the following items to install and connect the Ammasso adapter:

- ESD-preventative wrist strap
- Ammasso 1100 High Performance Ethernet Adapter
- Cat 5E cable terminated with an RJ-45 connector

2.3.3 Adapter Insertion Procedure

To install the Ammasso 1100 High Performance Ethernet Adapter into a server:

1. Power off the server and unplug the power cord.
2. Remove the cover from the server.
3. Put on an ESD-preventative wrist strap, and attach it to an unpainted, grounded metal surface.
4. Select an appropriate PCI-X connector slot in the server.

NOTE: Attaching more than one card to the PCI-X bus on your computer may degrade the Ammasso 1100 Ethernet Adapter's performance. Before selecting a slot, be sure to read the "Choosing a Slot for Installation" section above.

5. Remove the blank filler panel where you plan to install the adapter.

CAUTION: *To prevent ESD damage, handle the adapter only by its edges, and use an ESD-preventative wrist strap or other grounding device.*

6. Remove the Ammasso 1100 High Performance Ethernet Adapter from its anti-static packaging.
7. Align the adapter with the connector slot. Using the top of the adapter, and without forcing it, push the adapter gently but firmly into place until you feel its edge mate securely with the connector slot. Make sure that the adapter aligns so that you can see the port through the back of the computer.
8. Use a screw to attach the adapter to the chassis (see system documentation for details).
9. Replace the computer cover.
10. Attach the RJ45 cable to the port on the adapter and plug the other end of the cable into a Gigabit Ethernet switch port.
11. Plug the computer into AC power, turn it on, and reboot.
12. Login to the system as `root` and use the `lspci`(8) command to verify that the system recognizes the Ammasso 1100 card. The first few characters of the output are system specific.

```
# lspci -n | grep -i 18b8
02:02.0 Class 0b40: 18b8:b001 (rev 01)
```

13. Verify that the card has established link connectivity by observing the LEDs on the RJ45 connector. The Green "Link" LED should be constantly illuminated once link is established. The Yellow "Activity" LED will flash when there is network traffic. *The Ammasso 1100 will not establish link if connected to a 10/100 Ethernet switch as it requires a port speed of 1 gigabit.*
14. Proceed to Chapter 3, Adapter Software Installation.

3 Adapter Software Installation

This section provides information on installing the Ammasso adapter software.

3.1 System Software Requirements

System requirements for installing the software:

- Intel IA-32 and EM64T or AMD Opteron 32/64 compatible platform
- Disk Space – at least 50 MB of free disk space.
- Linux Operating System – see release notes for tested distributions
- C compiler must be installed in order to use MPICH
- Kernel source RPM installed, configured, and built (see section 3.4 for distribution specific details)

3.2 Overview

`AMSO1100.tgz` is the base package, which includes microcode, libraries and drivers. This must be installed to utilize the Ammasso 1100 adapter.

`AMSO_MPICH.tgz` is the MPICH 1.2.5 package and requires the AMSO1100 package. This package must be installed to enable MPI applications over the Ammasso 1100 adapter.

`AMSO_DAPL.tgz` is the DAPL package and requires the AMSO1100 package. This package must be installed to enable DAPL applications over the Ammasso 1100 adapter.

These packages must be built and installed on each machine in your cluster. The following instructions detail how to install on one node. For cluster-wide deployment on multiple nodes, please also see chapter 6 – Cluster Installation.

3.3 Installing the Adapter Software Package

NOTE: If updating your machines from a previous release, see the HOWTO_UPDATE.txt file for details on how to update the Ammasso 1100 hardware image if necessary and how to remove a previous installation.

1. Unarchive the `AMSO1100.tgz tar` file into a working directory.

```
# cd <work_dir>
# tar -zxf <path_to>/AMSO1100.tgz
```

2. Change into the `AMSO1100` directory, make the drivers, and direct the output to a logfile.

```
# cd ./AMSO1100
# make install 2>&1 | tee make_install.logfile
```

3. Answer the configure questions for your specific needs. The questions asked are described below.

   Q1: `Where would you like your software installed? [/usr/opt/ammasso]`

Enter the `AMSO1100` install directory. This is where the commands, libraries, and drivers will be installed.

Q2: `Where would you like your config data stored? [/usr/opt/ammasso/data/<hostname>]`

This is where the `rnic_cfg` file for the Ammasso installation will be stored. This file contains the network settings for the Ammasso adapter.

Q3: `Would you like to configure interfaces of the Ammasso 1100 adapter? (y or n) [y]`

If you have an `rnic_cfg` file from a previous installation, answer `'n'` to this question in order to retain that file. Answering `'y'` allows you to configure both of the IP addresses for the adapter, its network masks, gateway addresses, *etc*. for this system.

At this point you have successfully installed the AMSO1100 package on your system. If you are deploying the Ammasso software on many machines in a cluster, please see chapter 6 for more details. The various commands, libraries, and drivers are installed in the directory chosen in step 3.

The commands are linked into `/usr/bin`. The libraries are linked into `/usr/lib` or `/usr/lib64` as required. The man pages are linked into `/usr/share/man`. The startup configuration script is copied into `/etc/init.d` and linked appropriately in `/etc/rc*.d`. The Ammasso configuration file is copied into `/etc/ammasso.conf`.

NOTE: You must reboot your system for the drivers to be installed into the active (running) kernel.

After you reboot, you can verify that that the drivers are loaded by using the `ccons(8)` command to query the running firmware for the version number:

```
# ccons 0 vers
Cluster Core 1.2.1
#
```

3.3.1 Makefile Targets

The AMSO1100 package top-level `Makefile` supports the following targets:

`all`:
: Configures and builds AMSO1100 (default target).

`config`:
: Configures the AMSO1100 source tree for building. The result is a file `Config.mk` that contains various environment variables needed to build the product.

`build`:
: Builds AMSO1100 -- depends on `config` target.

`install`:
: Installs AMSO1100 -- depends on `build` target.

`uninstall`:
: Uninstalls AMSO1100 if it is installed.

`clean`:
: Deletes (cleans) the build objects from the source tree.

`binary`:
: This will put together a binary that can be put on other machines with identical setups. The resulting file `ammasso1100.bin` can be executed on each identical system in a cluster to install and configure the AMSO1100 package.

3.3.2 Makefile Configuration Variables

The configuration file `Config.mk` has the following variables:

`GCC_PATH`
The command name of the compiler program to use and optionally its associated pathname.

`LD_PATH`
The command name of the loader program to use and optionally its associated pathname.

`PLATFORM`
The target build platform. Possible values are `x86_32` and `x86_64`.

`KERNEL_SOURCE`
The pathname to the kernel source tree for the kernel you are running.

`KERNEL_CODE`
The release string for the kernel as returned by the `uname -r` command.

`O`
Path to alternate kernel build output directory.

`BUILD_32LIBS`
If the AMSO1100 package is configured on a 64 bit distribution, this variable will be set to `'y'` if 32 bit libraries should be built in addition to 64 bit libraries.

Here is a sample `Config.mk` created by the `'make config'` rule:

```
#
# This build configuration was automatically generated.
#
GCC_PATH=/usr/bin/gcc
LD_PATH=/usr/bin/ld
PLATFORM=x86_64
KERNEL_SOURCE=/lib/modules/2.4.21-20.ELsmp/build
KERNEL_CODE=2.4.21-20.ELsmp
O=
BUILD_32LIBS=y
```

3.4 Distribution Specific Build Details

The following are known issues with building on common distributions. Please refer to the Ammasso support website (www.ammasso.com/support) for an up to date list of issues. The list of packages provided below will ensure that the system will be able to take advantage of all the Ammasso 1100 features – such as being able to support 32-bit MPI applications on 64-bit platforms.

3.4.1 RedHat

The following section lists RedHat distribution specific details. While the exact keystrokes may vary slightly from release to release, the following are offered as guidelines for these distributions.

3.4.1.1 RedHat Package Selection

On 32-bit platforms, ensure that both `Development Tools` and `Kernel Development` packages are selected from `redhat-config-packages` menu. These allow you to build the Ammasso driver software.

On 64-bit platforms, select the following packages in the `System` group under `redhat-config-packages` menu: `Development Tools`, `Kernel Development`, `Legacy Software Development`, and `Compatibility Arch Support`. In addition, under the `Development Group` menu select the following: `Compatibility Arch Development Support` and `Legacy Software Development`. Installing these packages will allow both 32-bit and 64-bit MPI applications to run on the 64-bit installed system.

3.4.1.2 RedHat Kernel Source Tree Preparation

First ensure the system has a clean source tree by doing a `make mrproper`:

```
# cd /usr/src/linux
# make mrproper
```

Next, edit the `Makefile` so that the version value matches the running kernel. By default, the variable `EXTRAVERSION` includes the string `custom`. Change this variable to match the running kernel. The running kernel version can be found using `uname -r`. For example, modify `-15.ELcustom` to `-15.ELsmp`.

After that, initialize the `.config` file for your system. This can be accomplished by copying the `config` file from `/boot` and doing a `make oldconfig`.

```
# cd /usr/src/linux
# cp /boot/config-`uname --kernel-release` /usr/src/linux/.config
# make oldconfig
```

Finally, execute the kernel dependent target to update configuration files and rebuild the source dependencies.

For 2.4 Kernels:

```
# make dep
```

For 2.6 Kernels:

```
# make prepare
```

3.4.2 SuSE

The following section lists SuSE distribution specific details. While the exact keystrokes may vary slightly from release to release, the following are offered as guidelines for these distributions.

3.4.2.1 SuSE Package Selection

During the system installation, ensure that the following packages are selected to be installed: `ssh` and/or `rsh`, `make`, `g77`, `gcc-c++`, `glibc` (both the `shared libs` and `glibc-devel` packages), and `kernel` (verify `kernel-source` is selected and `kernel-syms` if available).

When installing a 64-bit system, add the following `glibc` packages: `glibc 32-bit shared libs` and `glibc 32bit devel packages`.

3.4.2.2 SuSE Kernel Source Tree Preparation

First ensure the system has a clean source tree by doing a `make mrproper`:

```
# cd /usr/src/linux
# make mrproper
```

Next, edit the `Makefile` so that the version value matches the running kernel. By default, the variable `EXTRAVERSION` includes the string `custom`. Change this variable to match the running kernel. The running kernel version can be found using `uname -r`. For example, modify `-15.ELcustom` to `-15.ELsmp`.

After that, initialize the `.config` file for your system. This can be accomplished with the following:

```
# cd /usr/src/linux
# make cloneconfig
# make oldconfig
```

Finally, execute the kernel dependent target to update configuration files and rebuild the source dependencies.

For 2.4 Kernels:

```
# make dep
```

For 2.6 Kernels:

```
# make prepare
# make
```

3.4.3 Kbuild

When compiling against 2.6 kernels, the `'kbuild'` style `Makefiles` allow for an `O=` option to specified. This option tells `kbuild` where to put the configured kernel files. If the kernel is configured with `O=`, then all external modules must be built with the same parameter. If your kernel source was built using the `O=` variable, you must specify make install `O=<path to kernel files>` when building the package.

Ammasso 1100 Commands and Utilities

Man pages are available for the following commands. A short description is included here.

`amso_cfg(8)`
Configure the adapter network settings based on the configuration file.

`amso_mode(8)`
Change operating mode of an Ammasso installation from `release`, `support` (debug), or `none`(off). This command is used by Ammasso support personnel to turn on/off debugging information.

`amso_setboot(8)`
Provide for automatic startup of an Ammasso installation.

`amso_stat(8)`
Return operational information of the current running Ammasso 1100 installation.

`amso_uninstall(8)`
Remove an Ammasso installation.

`ccflash2(8)`
Ammasso firmware update utility.

`cclog(8)`
Ammasso RNIC debug logging utility.

`cconfig(8)`
Ammasso RNIC IP address configuration command.

`ccons(8)`
The comand to connect to the Ammasso RNIC internal console.

`ccping(8)`
Ammasso RNIC ping command.

`ccregs(8)`
Command to dump Ammasso registers for debug.

`ccroute(8)`
Ammasso RNIC route configuration command.

`crash_dump(8)`
Dump a crashed RNIC to a file to enable Ammasso support personnel to help debug it.

`rnic_cfg(8)`
Describe the Ammasso 1100 configuration file and its variables.

3.5 Configuring the Ammasso 1100 Adapter

The Ammasso 1100 configuration file stores the necessary networking information for both the RDMA and Linux NetDev-style `ccilnet` interfaces. A single file is used to store the entire Ammasso 1100 configuration for a given system.

The configuration file is located in the directory specified during the installation. This defaults to:

```
<install_dir>/data/`hostname -s`/rnic_cfg.
```

For example:

```
/usr/opt/ammasso/data/fred/rnic_cfg
```

contains the configuration for the host named '`fred`'.

NOTE: This file can be located anywhere on your system. The configuration file /etc/ammasso.conf contains a variable that contains the path to the rnic_cfg file.

The `amso_cfg(8)` command is used to configure the running system based on the data found in the `rnic_cfg(8)` configuration file. The configuration file can be created initially via the AMSO1100 '`make install`' rule, but can also be created or modified with a traditional text editor such as `vi`(1) or `emacs`(1)

3.5.1 Configuration Entries

The `rnic_cfg` file entries have the following format:

```
function amso_[type]_[rnic]_[instance] {
    AMSO_IPADDR=[ipaddr]
    AMSO_MASK=[mask]
    AMSO_MTU=[mtu]
    AMSO_GW=[gw]
    AMSO_BCAST=[bcast]
}
```

Where the following fields contained within brackets mean:

`[type]` This is the type of entry being defined. Currently there are two valid types:

"ccil" -- This is the mode to define a legacy interface Ethernet network. The backend will use `ifconfig(8)` to configure the Linux netdev-style ccilnet interface.

"`rdma`" -- This is the mode to define an RDMA IP address. The backend will use Ammasso's own `cconfig`(1) command to manage this interface.

`[rnic]` This is the current RNIC number for which we are defining the address. This number starts counting from `0` (`zero`). Note this number is here for future capability and must always be `0` for the 1.2 Update 1 release.

`[instance]` This is the instance number for a given definition. This number starts counting from `0`. Each instance is another IP address, netmask, etc. definition for a given RNIC instance. This allows having multiple IP addresses per RNIC.

`[ipaddr]` This is the specific IP address in network 'dotted quad' notation to use for this interface.

`[mask]` This is the network mask (`netmask`) of the specific configuration.

`[mtu]` This is the maximum transmittable unit (MTU) or frame size for the interface. This is not required. If it is not specified, Ammasso will set the MTU to the default value of 1500.

`[gw]` This is the network gateway IP address for this interface. This is specified in network `dotted quad' notation. It is optional and if none is specified, Ammasso will not configure a default gateway.

`[bcast]` This is the broadcast address for the network. It is given in network `dotted quad' notation. Its value is optional and if none is specified, the broadcast address is deduced from the `[ipaddr]` and `[mask]`. This field will be ignored if defined within an "`rdma`" `[type]`.

3.5.2 Sample Configuration File

NOTE: Since the configuration file is in Bourne shell script syntax, you can use the "#" comment character. Any entries that are not needed can be commented out or removed. Both the RDMA and CCIL addresses can be specified in one configuration file.

The following example shows one separate IP address for both the RDMA and the CCIL interfaces. There is only one adapter and one instance of each. *Note that the RDMA and CCIL addresses must never be identical.*

```
function amso_rdma_0_0 {
    AMSO_IPADDR=10.10.10.2
    AMSO_MASK=255.255.255.0
}

function amso_ccil_0_0 {
    AMSO_IPADDR=192.168.1.2
    AMSO_MASK=255.255.255.0
}
```

3.6 Verifying the Adapter Software Installation

Once your system has been installed and configured with the Ammasso 1100 hardware and software, you can verify correct installation with the following procedures.

To verify CCIL configuration, you can use the `ifconfig(8)` command. The CCIL interface is called ccilnet0.

```
# /sbin/ifconfig ccilnet0

   ccilnet0  Link encap:Ethernet  HWaddr 00:0D:B2:00:07:B2
             inet addr:10.40.48.52  Bcast:10.40.63.255  Mask:255.255.240.0
             inet6 addr: fe80::20d:b2ff:fe00:7b2/64 Scope:Link
             UP BROADCAST RUNNING  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:0 (0.0 b)  TX bytes:460 (460.0 b)
             Interrupt:28 Memory:ffffff00003b0000-ffffff00003b4000
```

To verify RDMA configuration, you can use the `cconfig(8)` command.

```
# cconfig 0

   RNIC Index 0:
           addr 10.40.32.52, mask 255.255.240.0 MTU 1500
```

To verify ccilnet connectivity, use the `ping`(8) command. In this example, node-A has ccilnet IP address 10.40.48.52 and node-B has ccilnet IP address 10.40.48.53. You can `ping` the ccilnet IP address from any other host on your network.

```
# /bin/ping 10.40.48.52
   PING 10.40.48.52 (10.40.48.52) 56(84) bytes of data.
   64 bytes from 10.40.48.52: icmp_seq=1 ttl=64 time=0.082 ms
```

```
# /bin/ping 10.40.48.53
   PING 10.40.48.53 (10.40.48.53) 56(84) bytes of data.
   64 bytes from 10.40.48.53: icmp_seq=1 ttl=64 time=0.889 ms
```

To verify RDMA connectivity, use the `ccping(8)` command. You can only use `ccping` to generate a response from a remote RDMA IP address. You will need two machines with Ammasso hardware and software installed in order to verify RDMA connectivity. In this example, node-A has RDMA IP address 10.40.32.52 and node-B has RDMA IP address 10.40.32.53. The first example shows that issuing the `ccping` command to the local RDMA address will not result in a reply. The second example shows that issuing the `ccping` command to a remote RDMA address will result in a reply.

```
# ccping 0 10.40.32.52
   pinging 10.40.32.52 via TCP SYN to port 1
   10.40.32.52 no answer

# ccping 0 10.40.32.53
   pinging 10.40.32.53 via TCP SYN to port 1
   10.40.32.53 is alive
   10.40.32.53 is alive
```

3.7 Removing an Adapter Software Installation

Use the `amso_uninstall(8)` command to remove the AMSO1100 installation:

```
# amso_uninstall
Removing Ammasso 1100 driver
Are you sure you want to continue? (y or n) [n] y
The Amso1100 software installed in /usr/opt/ammasso has been removed.
#
```

4 The Ammasso MPI Library

4.1 Overview

MPICH is a portable implementation of the Message Passing Interface (MPI) standard. Currently the Ammasso 1100 supplies and supports the MPICH version 1.2.5 of MPI from Argonne National Labs. The Ammasso MPICH implementation is a fully supported port of MPICH over Ammasso's RNIC verbs interface designed to take advantage of the Ammasso 1100's low latency and high throughput. Additional information, full documentation, and manual pages for MPICH are available on the MPICH web site at:

http://www-unix.mcs.anl.gov/mpi/mpich/

Note that other MPI implementations have been qualified and tested to run over top of the Ammasso RNIC exploiting the advantages it provides. See the specific MPI vendor's web site as well as Ammasso' support web site for details on those implementations.

The following section provides some basic information about leveraging the Ammasso adapter with your MPI application using the Ammasso MPI implementation. Installation, configuration, and tuning examples are supplied. The information provided is for reference, as there may be multiple ways to accomplish these tasks, depending on each development environment.

4.1.1 Compiler Support

The Ammasso 1100 is designed to work with the standard compiler suites available in the Linux community. We have tested our MPI implementation with the GNU C and the F77 suites that are bundled with the traditional Linux distributions such as Red Hat and SuSE. We have also tested our MPI implementation with the Intel Version 7x and 8x C/C++ and F90 suites. As we progress, other compilers may be added to our test suite. Please see the Ammasso support web site for details and specifics.

4.2 Installation

NOTE: In order to compile the Ammasso 1100 MPICH 1.2.5 Package, the AMSO1100 package must be installed and built. If you are updating from a previous release, please see the HOWTO_UPDATE.txt file for information on removing software from previous releases.

1. Unarchive the files in AMSO_MPICH.tgz into a working directory with:
```
# cd <work_dir>
# tar -zxf <path_to>/AMSO_MPICH.tgz
```

2. Change into the AMSO_MPICH directory, build Ammasso's MPICH implementation, and capture the output to a logfile with:
```
# cd ./AMSO_MPICH
# make install 2>&1 | tee make_install.logfile
```

3. Answer the configure questions for your specific needs. Below is a description of the questions asked.

Q1: `Enter the AMSO1100 build path:`

This is the full path to the AMSO1100 source code. Ammaso's MPICH implementation needs access to some files distributed in the AMSO1100 directory to correctly compile. This defaults to ../AMSO1100.

Q2: `Base directory to install mpich [/usr/opt/ammasso]:`

This is the directory Ammasso's MPICH implementation will be installed into.

Q3: `Enter path to c compiler [/usr/bin/gcc]:`

This is the path to the C compiler that will be used to build MPICH C programs.

Q4: `Enter path to c++ compiler [/usr/bin/g++]:`

This is the path to the C++ compiler that will be used when building C++ MPICH programs.

Q5: `Enter path to fortran 77 compiler (enter 'none' to skip) [/usr/bin/g77]:`

This is the path to the FORTRAN 77 compiler that will be used when building FORTRAN 77 MPICH programs.

Q6: `Build shared libraries (y or n)? [no]`

Enabling this option builds shared libraries (`.so` files) for use by applications. If this is left to the default of 'n', only the static libraries (`.a` files) are built.

Q7: `Enter path to remote shell [/usr/bin/rsh]`

Argonne, MPICH, from which Ammasso MPICH derives, assumes the use of the BSD `rsh(1)` command, hence Ammasso's choice to leave that as the default. However, best security practices recommend a stronger system; such as Secure Shell and Ammasso recommends sites consider its use.

This is the full pathname to the command that you want to use to launch applications on cluster nodes. The default Ammasso MPICH will use the `rsh(1)` command. This is typically installed as `/usr/bin/rsh`.

The *Secure Shell*, or `ssh(1)` command is traditionally installed as `/usr/bin/ssh`. A properly configured Secure Shell system will provide the needed mechanism. No matter what remote shell is chosen, Ammasso's MPICH requires that programs executed via the remote shell operate without the need for the user to enter a password.

Q8: `Build mpich using a FORTRAN 90 compiler (y or n)? [no]`

By default, Ammasso's MPICH searches only for a FORTRAN 77 compilation suite. Standard Linux distributions install GNU's F77 to `/usr/bin/f77`. If the user has installed an optional FORTRAN 90 compilation suite and wishes Ammasso's MPICH to use it as well, the user should reply 'y' to this question.

When the FORTRAN 90 option is selected, the default for the MPICH libraries produced is to move the F77 and F90 routines into separate libraries noted by F77 or F90 in the name. Some applications may expect to link in only one library with combined C and F77 routines. If this is the case, modify the `F90ARGS` line in the `<work_dir>/AMSO_MPICH/mpich-1.2.5/conf` file from:

```
F90ARGS="-with-flibname=f77mpich -with-f90libname=f90mpich"
```

to:

```
F90ARGS="-with-f90libname=f90mpich"
```

Q9: `Enter path to FORTRAN 90 compiler`

If the user said 'y' to Q5 then the user is prompted for the full pathname to the installed FORTRAN 90 compilation suite. The install procedure will check whether the environment variable LD_LIBRARY_PATH points to the lib directory of the FORTRAN 90 compiler. If it does not, the install will fail.

Q10: `Build 32-bit mpich (y or n)? [no]`

This question will only appear when Ammasso's MPICH implementation is built on a 64-bit machine. By default, Ammasso builds only one version of MPICH on any machine. If the user wishes to have a 32-bit and 64-bit version of Ammasso MPICH compiled and installed, please answer `'y'` here. A case when the 32-bit MPICH option would be chosen would be to support a legacy 32-bit application for which no sources are available to recompile. Another case would be in a cluster environment which has both 32-bit and 64-bit systems.

At this point Ammasso MPICH is configured and will begin the compilation and installation process. The files are installed into the `<installation_dir>/mpich-1.2.5` directory. If you are on a 64-bit machine and have taken the option to build the 32-bit version of Ammasso MPICH, there will also be `<installation_dir>/mpich-1.2.5-32`.

4.2.1 Makefile Targets

The MPI `Makefile` supports the following targets:

`all:`
: Configures and builds Ammasso MPICH, this is the default target for `make`.

`config`:
: Configures the Ammasso MPICH source tree, this rule creates the file `Config`.

`build:`
: Builds Ammasso MPICH and depends on `config` target.

`install:`
: Installs Ammasso MPICH and depends on `build` target.

`uninstall:`
: Uninstall Ammasso MPICH if it is installed.

`clean:`
: Cleans the Ammasso MPICH source tree of any previously built objects.

`binary:`
: This will put together a binary that can be put on other machines with identical setups. The resulting file `image-mpich-1.2.5.bin` can be executed on each identical system in a cluster to install and configure the AMSO_MPICH package. A 32-bit binary file called `image-mpich-1.2.5-32.bin` will also be generated if you are running on a 64-bit machine and have chosen to build the 32-bit version of Ammasso MPICH.

4.2.2 Makefile Configuration Variables

The configuration file `Config` has the following variables:

`CC`
This is the path to the C compiler that will be used to build MPICH C programs.

`CPP`
This is the path to the C++ compiler that will be used when building C++ MPICH programs.

`FC`
This is the path to the FORTRAN 77 compiler that will be used when building FORTRAN 77 MPICH programs.

`STARCORE`
This is the path to the AMSO1100 source tree. It must be an absolute path.

`LDFLAGS`
This is a list of load flags passed to the loader. The most common flags found in this are of the form `"-L/path/to/library"`. The contents of this variable are white space delimited.

INSTALL_DIR
This is the full pathname into which Ammasso MPICH will be installed. It must be an absolute pathname not relative.

RSHCOMMAND
This is the full pathname to the remote shell command. It must be an absolute path pathname not relative.

ENABLE_SHAREDLIB
This is set to 'yes' if we enable building of shared libraries. Otherwise, it's set to 'no'.

F90
This is the full path to the FORTRAN 90 compiler. It must be an absolute pathname not relative.

F90FLAGS
This is a list of flags passed to the FORTRAN 90 compiler. This variable is set by the install utility automatically based on which FORTRAN 90 compiler is selected.

F90_LDFLAGS
This is a list of load flags passed to the FORTRAN 90 compiler. This gets set by the install utility automatically based on the FORTRAN 90 compiler selected.

PLATFORM
This variable determines the platform on which to build Ammasso MPICH. The currently supported options are native and x86_32. The latter is only necessary if you wish to build a 32-bit version of Ammasso MPICH on a 64-bit version of the native OS.

CUSTOM_CFLAGS
This is a list of flags passed to the C compiler. This gets set by the install utility based on which C compiler is selected.

BPROC
This variable specifies whether to use mpirun_broc if running on a Scyld cluster.

4.3 Locating the MPI Libraries and Files

Most of the standard MPICH directories can be found in the directory in which you unarchived the AMSO_MPICH.tgz package.

```
<work_dir>/mpich-1.2.5
```

The source code for the MPICH driver for the Ammasso adapter can be found in

```
<work_dir>/mpich-1.2.5/mpid/iwarp
```

The libraries and files associated with using MPICH and the Ammasso 1100 are located in a directory within the Ammasso installation environment:

```
<install_dir>/mpich-1.2.5
```

If you chose to install a 32b version of MPICH on a 64b system, there will be an additional directory:

```
<install_dir>/mpich-1.2.5-32
```

Additionally, the standard MPICH examples directory is located in:

```
<install_dir>/mpich-1.2.5/examples
```

For the example used below, the `cpi.c` program is found in this directory. The `cpi` program will calculate the value of Pi.

4.4 Compiling and Linking Applications

The directory `<install_dir>/mpich-1.2.5/bin` contains a number of scripts used for compiling and linking user applications. For example, C applications can be built using the "`mpicc`" command script. These scripts set up and use all the correct settings for the standard Ammasso version of MPICH.

As an example, to compile and link the `cpi.c` program, the following steps need to be performed on each system that will be used for running the program. Since it is important that the path to the application is the same on all machines, most sites use a distributed file system such as NFS, Lustre *etc.* to provide a single file name space for all files used by MPI.

Create an object file using the Ammasso MPICH compilation script "`mpicc`" provided in the MPICH `bin` directory listed above by using the following command, note that this assumes that the `MPICH bin` directory is in the user's path , i.e. the user already added the following to a `.cshrc, .profile,` or `.bashrc` file:

C Shell users should add the following command line to `.cshrc` file:

```
# set path = ( <install_dir>/mpich-1.2.5/bin $path )
```

Bourne/Korn shell or bash users should add the following command lines to `.profile` or `.bashrc` file respectively:

```
# PATH=<install_dir>/mpich-1.2.5/bin:$PATH
# export PATH
```

Compile the program by using the following command:

```
# mpicc -c cpi.c
```

Link it by using the following command (note that in this case, it requires the math library):

```
# mpicc -o cpi cpi.o -lm
```

NOTE: A `Makefile` is provided in the examples directory. With this `Makefile`, one just needs to run make to compile and link the program. This `Makefile` can be used as a starting point for building other MPICH applications.

4.4.1 Creating a MPI Cluster `Machines` File

Before you can run an MPI program on your cluster, you need to create a `machines` file to tell MPI about the cluster configuration. The location of the `machines` file is:

```
<install_dir>/mpich-1.2.5/share/machines.LINUX
```

Or for a 32-bit install on a 64-bit machine

```
<install_dir>/mpich-1.2.5-32/share/machines.LINUX
```

Using a text editor, create the appropriate machines file, containing a list of all the host machines that you will be using, one per line. For example:

```
# cd <install_dir>/mpich-1.2.5/share
# cat machines.LINUX
hostA         # Comments are also allowed
hostB
```

Each host listed must either be a valid IP address or a host name that returns a valid IP address from the DNS.

NOTE: the TCP/IP address used for MPI control should not be confused with the RDMA IP Address which is used by MPI to move MPI data. The TCP/IP address traditionally would correspond to the `AMSO_IPADDR` *found in the* `amso_ccil_0_0` *function in the* `rnic_cfg` *file or a TCP/IP address associated with another NIC (e.g. eth0).*

The Ammasso MPICH implementation does not wrap the machine file hostnames. If you specify a command line to run with four processes (`-np 4`), the `machines.LINUX` file must have four host processors listed. Also, the Ammasso MPICH implementation does not start an MPICH process on the local host by default. Each node on which you wish to run the MPICH job must be specified in the `machines.LINUX` file.

It is possible to run multiple instances of a program on the same host. Either enter that hostname into the file multiple times, or add a colon and a number on the corresponding line in the machines file. For example:

```
# cd <install_dir>/mpich-1.2.5/share
# cat machines.LINUX
hostA:2             # Two instances on this node
hostB:4             # Four instances on this node
```

NOTE: The maximum number of instances supported on a single node is four.

NOTE: When running multiple processes of a program on a single host, the processes communicate using host memory, not the Ethernet. There is an upper limit on the size of messages that can be sent between processes on the same host which depends on the number of instances on the host and the total amount of physical memory. On a system with 1G of physical RAM, when running 2 instances on the same node, the upper limit is 32M bytes; when running 3 instances on the same node, the upper limit is slightly more than 16M bytes; when running 4 instances on the same node, the upper limit is slightly more than 8 M bytes. See the Ammasso 1100 Technical Notes at

http://www.ammasso.com/support

for specific details about how this is determined, and how a user can modify these limits.

4.4.2 Remote Startup Service

AMSO_MPICH makes use of standard `rsh`(1) and `ssh`(1) remote execution services for startup of remote MPI processes. This requires that each cluster node be accessible via `rsh(1)` or `ssh(1)` without the need to enter a password. You can verify that this is the case by issuing a simple command, *e.g.*

```
# rsh hostA hostname
hostA
```

The default remote execution service is selected during install. To change the remote execution service dynamically, set the environment variable P4_RSHCOMMAND. For example, using the bash shell:

```
# export P4_RSHCOMMAND=/usr/bin/ssh
```

4.5 Verifying MPI Installation

Once an application has been compiled and linked with the Ammasso MPICH driver, it can be run like any other MPI application.

The example `cpi.c` program that was compiled and linked above on 2 systems can be used to verify that the MPI installation is correct. The following commands may be used:

```
# cd <install_dir>/mpich-1.2.5/examples
# <install_dir>/mpich-1.2.5/bin/mpirun -np 2 ./cpi
```

Here `-np` specifies the number of processes with which to run the test. In this example two (2) processes are specified. This number must be less than or equal to the number of lines in the machines file created above.

The output from running the above program produces the following output:

```
       Process 1 on hostB
       Process 0 on hostA
pi is approximately 3.1416009869231241, Error is 0.0000083333333309
       wall clock time = 0.000263
```

4.6 Removing the Ammasso MPI Installation

Use the `mpich_uninstall` command to remove the AMSO_MPICH installation. This command will not remove `machines.*` files in the `<install_dir>/mpich-1.2.5/share` directory. These files are left on the system for use in the future if needed.

```
# . /etc/ammasso.conf
# $INSTALL_DIR/mpich-1.2.5/mpich_uninstall
Uninstall mpich found at /usr/opt/ammasso/mpich-1.2.5 (y or n)? [no] y
The following files were saved:
/usr/opt/ammasso/mpich-1.2.5/share/machines.sample
/usr/opt/ammasso/mpich-1.2.5/share/machines.LINUX
#
```

If a 32b library exists on a 64b system, remove that installation as well:

```
# . /etc/ammasso.conf
# $INSTALL_DIR/mpich-1.2.5-32/mpich_uninstall
Uninstall mpich found at /usr/opt/ammasso/mpich-1.2.5-32 (y or n)? [no] y
Mpich in /usr/opt/ammasso/mpich-1.2.5-32 has been removed.
The following files were saved:
/usr/opt/ammasso/mpich-1.2.5-32/share/machines.sample
/usr/opt/ammasso/mpich-1.2.5-32/share/machines.LINUX
#
```

4.7 Ammasso MPI Tunable Parameters

The following parameters can be used to tune the maximum amount of memory that will be consumed by the AMSO_MPICH RDMA driver. Memory consumed by the driver is locked down, or pinned, and not available for application or even operating system use, so at times it can be necessary to adjust these parameters to limit how much memory the driver will lock down.

NOTE: This also affects the performance of the driver, and overly reducing these limits can cause extremely poor performance. Conversely, setting these too high can cause consumption of all memory on the system. Caution should be used when adjusting these parameters.

For the purpose of this discussion, four variables are defined; Node Count, (NC), Local Process count (LP), Number of Processes (NP), and the Remote Process count (RP).

NC is defined as the number of nodes used in an MPI run.

LP is defined as the number of *local* processes on each compute node. We assume, in this discussion, that each compute node runs the same number of MPI processes.

NP is the *total* number of processes in the MPI run. Assuming each node runs the same number of processes, NP is computed as:

```
NP = (NC * LP)
```

RP is defined as the number of *remote* processes relative to any given MPI process in an MPI run. The remote process count is important because each MPI process connects to *every* remote process in the run using an RDMA Queue Pair. And each RDMA Queue Pair consumes memory.

Assuming each node runs the same number of processes, RP is computed as:

```
RP = (NC * LP) - LP.
```

For example:

NC	LP	NP	RP
16	1	16	15
32	2	64	62
64	3	192	189

The main data structure used to RDMA data between processes is called a *vbuf*. Any given IO operation consumes some number of *vbufs* based on the size of the IO operation. Each *vbuf* can convey 8256 bytes of application data. The size of the *vbuf* structure including space for payload is 8360 bytes. Thus, the value 8360 is used below to compute memory utilization.

4.7.1 VIADEV Environment Variables

NOTE: Shell environment variables are used to allow tuning memory consumption by the MPI RDMA driver. These variables must be set in the user account used to execute the MPI run (EG: the `.bashrc` file). Further, the values must be identical on each node in the cluster. Otherwise, the run will fail.

The following table lists these variables, their default values, and a short description.

Environment Variable Name	Default value	Description
`VIADEV_NUM_RDMA_BUFFERS`	NP <= 4, 1024 4 < NP <= 8, 512 8 < NP <= 16, 256 16 < NP <= 128, 128 128 < NP <= 256, 64 256 < NP <= 512, 32 512 <= NP, 24	The number of RDMA Write buffers to use *per* RDMA connection.
`VIADEV_RQ_DEPTH`	NP <=64, 240 NP > 64, 120	The Receive Queue depth for each RDMA Queue Pair.
`VIADEV_SQ_DEPTH`	NP <= 64, 256 NP > 64, 128	The Send Queue depth for each RDMA Queue Pair.

VIADEV_MAX_RENDEZVOUS	209715200 Bytes	The maximum amount of user buffer memory that will be locked down at any point in time for zero-copy IO.

4.7.1.1 VIADEV_NUM_RDMA_BUFFERS

This parameter specifies how many RDMA Write buffers will be setup per connection. The amount of memory consumed *per process* for RDMA Write buffers is:

```
2 * VIADEV_NUM_RDMA_BUFFERS * 8360 * RP
```

Thus, by default, on a 16 node run with 1 process on each node (NC=16, LP=1, NP=16, RP=15), the memory used for RDMA Write buffers would be 2 * 256 * 8360 * 15 = 64204800 bytes (or 61.23 MB)

4.7.1.2 VIADEV_RQ_DEPTH

This parameter specifies the Receive Queue (RQ) depth for each RDMA Queue Pair (QP). This depth acts as a flow control mechanism for message passing between MPI processes across the fabric. The amount of memory consumed *per process* for the RQ is:

```
VIADEV_RQ_DEPTH * 8360 * RP
```

So by default, on a 16 node run with 1 process on each node (NC=16, LP=1, NP=16, RP=15), the memory used for receive buffers would be 240 * 8360 * 15 = 30096000 bytes (28.70 MB)

4.7.1.3 VIADEV_SQ_DEPTH

This parameter specifies the Send Queue (SQ) depth for each RDMA Queue Pair (QP). This depth acts as a flow control mechanism between the MPI application and the local RNIC adapter.

Each SQ entry, when doing a particular IO operation (RDMA SEND), will consume one vbuf to describe the application data being sent. Assuming all SQs on all QPs are full of these SENDS, the amount of memory consumed *per process* is:

```
VIADEV_SQ_DEPTH * 8360 * RP
```

So by default, on a 16 node run with 1 process on each node (NC=16, LP=1, NP=16, RP=15), the maximum memory consumed for full SQs would be 256 * 8360 * 15 = 32102400 (30.62 MB).

4.7.1.4 VIADEV_MAX_RENDEZVOUS

This parameter limits the amount of application buffers that the MPI RDMA driver will lock down for doing zero-copy RDMA IO. Zero-copy IO is only done if the IO request from the application is sufficiently large enough to warrant the overhead of buffer registration and rendezvous overhead.

Each MPI process is allowed to lock down up to VIADEV_MAX_RENDEZVOUS amount of application buffer memory. Once this limit is reached, IO buffers are evicted and unregistered on a "least recently used" basis.

The default value for this parameter is 209715200 bytes (200MB).

4.7.2 Tuning suggestions

Based on the default values, one can determine how much memory is being consumed by each MPI process in a run. Then the desired amount of memory should be determined based on how much memory your MPI application needs for doing computation. Given you can actually determine that, you can then adjust these parameters to reduce the amount of memory used by the MPI RDMA driver.

A few guidelines are offered here:

- Try reducing VIADEV_NUM_RDMA_BUFFERS first. The default for this parameter doesn't particularly scale well or evenly as your NP value increases.
- Try reducing VIADEV_MAX_RENDEZVOUS as a next step.
- Keep VIADEV_SQ_DEPTH >= VIADEV_RQ_DEPTH
- Try to avoid lowering VIADEV_RQ_DEPTH. It is probably safe to drop this to around 64, but avoid going lower if possible.

5 The Ammasso DAPL Library

5.1 Overview

The Ammasso 1100 DAPL release is composed of one source code tar package. `AMSO_DAPL.tgz` is the DAPL package. This package must be installed to enable both user and kernel mode uDAPL and kDAPL applications over the Ammasso 1100 adapter.

Currently, the Ammasso 1100 supports the version 1.2 uDAPL and kDAPL API specifications. Applications built against version 1.1 DAPL will need to be recompiled to run with the 1.2 DAPL libraries.

The following sections describe how to build and install the DAPL package.

5.2 Installation

NOTE: *The Ammasso 1100 DAPL Package requires a built and installed AMSO1100 package in order to build. If you are updating from a previous release, please see the HOWTO_UPDATE.txt file for information on removing software from previous releases.*

1. Untar `AMSO_DAPL.tgz` into a working directory.

```
# cd <work_dir>
# tar -zxf <path_to>/AMSO_DAPL.tgz
```

2. Change into the AMSO_DAPL directory, build dapl, and capture the output in a logfile.

```
# cd ./AMSO_DAPL
# make install 2>&1 | tee make_install.logfile
```

3. Answer the configure questions for your specific needs. Below is a description of the questions asked.

   Q1: `Enter the AMSO1100 build path`

   This is the full path to the AMSO1100 source code. DAPL needs certain files in this directory to compile correctly. This defaults to `../AMSO1100`

   Q2.a: `Do you want to load kdapl at boot time (y or n) [YES]?`

   Answer `'y'` if you want the `kdapl` module inserted into the running kernel at boot time. If you answer `'y'`, then the following question is also asked:

   Q2.b: `Do you want to load kdapltest at boot time (y or n) [YES]?`

   Answer `'y'` if you want the `kdapltest` modules inserted into the running kernel at boot time.

At this point you have successfully installed DAPL onto your system. The files are installed into the `<installation_dir>/dapl-1.2` directory.

5.2.1 Makefile Targets

The DAPL `Makefile` supports the following targets:

`all:`
Configures and builds DAPL (default target).

`config:`
Configures the DAPL source tree. This rule creates the file `Config`.

`build:`
Builds DAPL -- depends on `config` target.

`install:`
Installs DAPL -- depends on `build` target.

`uninstall:`
Uninstall DAPL if it is installed.

`clean:`
Cleans the DAPL tree of any previously built programs.

`binary:`
This will put together a binary that can be put on other machines with identical setups. The resulting file `image-dapl-1.2.bin` can be executed on each identical system in a cluster to install and configure the AMSO_DAPL package.

5.2.2 Makefile Configuration Variables

The configuration file `Config` has the following variables:

`STARCORE`
This is the path to the AMSO1100 source tree. It must be an absolute path.

`PLATFORM`
The target build platform. Possible values are `x86_32` and `x86_64`.

`LOADKDAPL`
This variable specifies whether to load the kdapl kernel module at boot time.

`LOADKDAPLTEST`
This variable indicates whether to load the kdapltest kernel module at boot time.

`KERNEL_CODE`
The release string for the kernel, as returned by the `uname -r` command.

KERNEL_SOURCE
The pathname to the kernel source tree for the kernel you are running.

O
Path to alternate kernel build output directory.

5.3 Configuring DAPL

The DAT registry file is created as part of the 'make install' process and is copied into /etc/dat.conf. The file is created with the Ammasso DAPL provider already registered, so no modifications are needed by default. If an /etc/dat.conf file already exists at install time, the Ammasso entry will be appended into the /etc/dat.conf file to allow multiple providers on one system.

It is optional whether the kdapl and kdapltest modules are loaded at boot time. The file <install_dir>/dapl-1.2/etc/kdapl.conf can be edited to enable kdapl and kdapltest to be loaded at boot time. Edit this file and modify the LOADKDAPL and LOADKDAPLTEST variables to `YES`. The default is `NO`, which prevents loading any kDAPL modules at system boot time.

5.4 Verifying DAPL Installation

Once the DAPL software has been installed, sample programs are available to verify the installation. These programs can be found in <install_dir>/dapl-1.2/bin. The sample programs are client/server programs. Two nodes are required in order to run these examples. One node must start the server program first then the second node can start the client program. Test scripts are available both for uDAPL and kDAPL.

5.4.1 uDAPL Installation Verification

For example, with two nodes, hostA and hostB, hostA starts the server program first:

```
# cd <install_dir>/dapl-1.2/bin
# ./srv.sh
Dapltest: Service Point Ready - ccil0
```

hostB can now start the client program and specify hostA's RDMA address as the address to connect to:

```
# cd <install_dir>/dapl-1.2/bin
# ./regress.sh 10.40.32.52
Dapltest: Service Point Ready - ccil0
Server Name: 10.40.32.52
Server Net Address: 10.40.32.52
DT_cs_Client: Starting Test ...
----- Stats ---- : 1 threads, 1 EPs
Total WQE        :   17543.85 WQE/Sec
Total Time       :       1.13 sec
Total Send       :       2.56 MB -       2.24 MB/Sec
Total Recv       :       2.56 MB -       2.24 MB/Sec
Total RDMA Read  :       0.00 MB -       0.00 MB/Sec
Total RDMA Write :       0.00 MB -       0.00 MB/Sec
```

```
DT_cs_Client: ========== End of Work -- Client Exiting
...
```

This test will take a few minutes to complete.

5.4.2 kDAPL Installation Verification

The kDAPL test scripts require that both the `kdapl` and `kdapltest` kernel modules are loaded. With two nodes, hostA and hostB, hostA starts the server program first:

```
# cd <install_dir>/dapl-1.2/bin
# ./ksrv.sh
Dapltest: Service Point Ready - ccil0
```

hostB can now start the client program and specify hostA's RDMA address as the address to connect to:

```
# cd <install_dir>/dapl-1.2/bin
# ./kregress.sh 10.40.32.52
Server Name: 10.40.32.52
Server Net Address: 10.40.32.52
DT_cs_Client: Starting Test ...
----- Stats ---- : 1 threads, 1 EPs
Total WQE        :   17543.85 WQE/Sec
Total Time       :       1.13 sec
Total Send       :       2.56 MB -       2.24 MB/Sec
Total Recv       :       2.56 MB -       2.24 MB/Sec
Total RDMA Read  :       0.00 MB -       0.00 MB/Sec
Total RDMA Write :       0.00 MB -       0.00 MB/Sec
DT_cs_Client: ========== End of Work -- Client Exiting
...
```

This test will take a few minutes to complete.

5.5 Removing the Ammasso DAPL Installation

Use the `dapl_uninstall` command to remove the AMSO_DAPL installation:

```
# . /etc/ammasso.conf
# $INSTALL_DIR/dapl-1.2/dapl_uninstall
Uninstall dapl found at /usr/opt/ammasso/dapl-1.2 (y or n)? [no] y
Dapl in /usr/opt/ammasso/dapl-1.2 has been removed.
```

5.6 Ammasso DAPL Compatibility Settings

The following shell environment variables are provided to enable compatibility with other DAPL provider libraries. A brief description of each is listed below.

5.6.1 CCAPI_ENABLE_LOCAL_READ

The IBTA Infiniband specification states that all memory regions have local read access implicitly (vol 1, sec 10.6.3.1). This is not true for the IWARP RDMA Verbs 1.0 Specification, which states that the consumer must explicitly enable local read access on memory regions. This can lead to application errors for DAPL applications currently running

on Infiniband DAPL providers that are ported to the Ammasso DAPL provider. The solution is to explicitly set local read access on memory regions which is valid for both IB and iWARP.

Ammasso provides a workaround for this issue: Set `CCAPI_ENABLE_LOCAL_READ`=1 in your environment before executing your uDAPL application. This will set local read privileges for your application implicitly when memory regions are registered.

5.6.2 DAPL_1_1_RDMA_IOV_DEFAULTS

With the release of the 1.2 version of the DAT API, new endpoint attributes have been defined to allow the consumer application to specify a maxmimum IOV depth for RDMA Read and RDMA Write DTO requests. Version 1.1 of the DAT API only specified the maximum IOV depth for SEND DTO requests. With the dapl-1.2 release from Ammasso, if a consumer application does _not_ specify these new attributes when creating a DAT endpoint, they will default to zero, thus disabling RDMA Read and Write DTOs on that endpoint.

To ease application migration from 1.1 to 1.2, an environment variable can be set to make these attributes default to the send maximum IOV depth. Set `DAPL_1_1_RDMA_IOV_DEFAULTS`=1 and these new attributes (max_rdma_write_iov and max_rdma_read_iov) will default to the SEND maximum IOV depth attribute (max_send_iov).

6 Cluster Installation

6.1 Introduction

The purpose of this chapter is to provide a sample install session for the Ammasso 1100 adapter software (AMSO1100), MPICH and DAPL packages. From the install on one node, the software is then deployed to several nodes across a cluster. The steps to install on an initial node in the cluster are different than those used for the follow on cluster nodes. Both procedures are documented. This document lists steps that are specific to SuSE 9.1 64 bit systems.

6.2 Steps on the Initial Build System

Full AMSO1100, MPICH and DAPL builds are only required for one node within a cluster provided all nodes within the cluster have the same Linux distribution, patch level and processor type. The node to be used for the build is referred to as the "initial build system" within this document. The remaining systems in the cluster will be referred to as the "cluster nodes".

6.2.1 Prepare the Kernel

1. If this is the first time this system has been used to build AMSO1100, prepare the kernel. This step needs to be done as the root user. A full make is needed for SuSE 9.1 since `<kernel_dir>/arch/x86_64/kernel/vmlinux.lds.s` is only created after a full make (`vmlinux.lds.s` is different from `vmlinux.lds.S`). It is not necessary to do full kernel installs on all distributions.

```
# uname -a
Linux blade-39 2.6.4-52-smp #1 SMP Wed Apr 7 01:58:54 UTC 2004 x86_64 x86_64
x86_64 GNU/Linux

# cd /usr/src

# ls -l
total 16
drwxr-xr-x   4 root root 4096 2004-11-06 23:38 .
drwxr-xr-x  16 root root 4096 2004-11-07 18:25 ..
lrwxrwxrwx   1 root root   14 2004-11-06 23:38 linux -> linux-2.6.4-52
drwxr-xr-x  20 root root 4096 2004-12-08 13:21 linux-2.6.4-52
drwxr-xr-x   7 root root 4096 2004-11-06 23:43 packages

# cd linux-2.6.4-52

# make mrproper
......
output cleaning the current configuration
......

# make cloneconfig
......
output configuring the kernel
......

# make
......
full kernel make, takes about 30 minutes
......
```

6.2.2 Install AMSO1100 and Build Binary for Cluster Deployment

1. Untar the AMSO1100 package on the build system. The unzip/tar-extract for this example was done in the `/tmp` directory but that is not a requirement.

```
# cd /tmp
# ls
.              AMSO1100.tgz  .ICE-unix         .X11-unix
..             gconfd-root         .qt               YaST2-02308-zPffTu
3Ddiag.Yp3238  hps.test            sysconfig-update
#
# tar zxf AMSO1100.tgz
# ls
.              AMSO1100      hps.test   sysconfig-update
..             AMSO1100.tgz  .ICE-unix  .X11-unix
3Ddiag.Yp3238  gconfd-root         .qt        YaST2-02308-zPffTu
#
```

2. Build and install the AMSO1100 driver. This step needs to be done as a user with root privileges. If a previous install has been done the make install will prompt whether it is okay to overwrite the potential installation at this location. The proper response is to answer 'y' unless you want to abort the install. Even though previous files may be deleted the data directory which contains configuration information will stay intact.

```
# cd /tmp/AMSO1100/
# make install
......
output of build, make takes approximately 5 minutes
......
Where would you like your software installed? (/usr/opt/ammasso)
Installing to /usr/opt/ammasso
The AMSO1100 software is already installed at /usr/opt/ammasso.
You may remove the software without destroying any of the node
configuration data (such as RDMA/CCIL IP addresses).
Is it ok to remove any previously installed files? (y or n) [y]
Re-installing to /usr/opt/ammasso
Saving stored data in /usr/opt/ammasso/data/app64-01
The Amso1100 software installed in /usr/opt/ammasso has been removed.
 * The installer has detected /usr/opt/ammasso/data/app64-01/rnic_cfg.
 * Please answer 'no' below to keep this configuration.
Configure interfaces of the Ammasso 1100 adapter? (y or n) [y]
Configure the RDMA network interface IP address settings? (y or n) [y]
Please enter the RDMA IP address (10.40.32.53):
Please enter the RDMA network mask (255.255.240.0):
Please enter the RDMA network gateway ():
Please enter the RDMA network MTU (1500):
Configure the legacy network interface IP address settings? (y or n) [y]
Please enter the legacy IP address (10.40.48.53):
Please enter the legacy network mask (255.255.240.0):
Please enter the legacy network Gateway ():
Please enter the legacy network MTU (1500):
Reboot the system to activate the AMSO1100 board and its software
#

# ls /usr/lib/libccil*
/usr/lib/libccil.a  /usr/lib/libccil.so

# ls -l /usr/lib64/libccil*
/usr/lib64/libccil.a  /usr/lib64/libccil.so

# find /etc -name "*amso1100" -print
/etc/init.d/rc2.d/S06amso1100
/etc/init.d/rc2.d/K16amso1100
/etc/init.d/rc3.d/S06amso1100
/etc/init.d/rc3.d/K16amso1100
```

```
/etc/init.d/rc5.d/S06amso1100
/etc/init.d/rc5.d/K16amso1100
/etc/init.d/amso1100

# ls /etc/ammasso.conf
/etc/ammasso.conf

# cat /etc/ammasso.conf
INSTALL_DIR=/usr/opt/ammasso
RNIC_CFG=/usr/opt/ammasso/data/app64-01
IS_INSTALLED=1
#

# ls /usr/opt/ammasso
.  ..  bin  data  fw  lib  lib64  man  release  support  scripts

# ls /usr/opt/ammasso/data
.  ..  app64-01  default

# ls /usr/opt/ammasso/data/app64-01
.  ..  mode  rnic_cfg

# tail -12 /usr/opt/ammasso/data/app64-01/rnic_cfg
function amso_rdma_0_0 {
   AMSO_IPADDR=10.40.32.53
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}
function amso_ccil_0_0 {
   AMSO_IPADDR=10.40.48.53
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}
```

AMSO1100 is now installed on the build system. Note the file `rnic_cfg` which is used to set the IP settings of the RNIC. This file will be used as a template for the cluster nodes. Before building MPICH and DAPL, the next step is to create the binary image file for installation on the cluster nodes.

3. Build a binary image for install on the cluster nodes. This step can also be done at a later time and after a reboot provided the AMSO1100 build directory is still present.

```
# make binary
......
output of make binary
......
Binary file ammasso1100.bin has been built. Please execute on your target
machine to install.
#

# ls
.   ammasso1100.bin  cset  Makefile  software
..  Config.mk        data  scripts   verbatim
```

`ammasso1100.bin` is a shell script and binary image for installation on cluster nodes. Save this file to a safe location that can be used to distribute to the cluster nodes.

```
# cp ammasso1100.bin /tmp
```

Build MPICH and DAPL before rebooting.

6.2.3 Install MPICH and Build Binary for Cluster Deployment

1. Use the `tar(1)` command to unarchive the MPICH package on the build system. The unzip/tar-extract for this example was shown using the `/tmp` directory but that is not a requirement. However placing the directory at the same location as the AMSO1100 build directory allows the configuration to automatically find it.

```
# cd /tmp
# ls
.               AMSO1100.tgz        .ICE-unix       YaST2-02308-zPffTu
..              AMSO_MPICH.tgz      .qt             3Ddiag.Yp3238   gconfd-
root     sysconfig-update    AMSO1100        hps.test            .X11-unix

# tar zxf AMSO_MPICH.tgz

# ls
.               AMSO1100.tgz         hps.test          .X11-unix
..              AMSO_MPICH          .ICE-unix         YaST2-02308-zPffTu
3Ddiag.Yp3238   AMSO_MPICH.tgz      .qt
AMSO1100        gconfd-root         sysconfig-update

# cd AMSO_MPICH/

# ls
.  ..  Makefile  mpich-1.2.5  mpich_cset  scripts
```

2. Configure MPICH for building. This set of instructions breaks the MPICH build/install into three separate make steps. If preferred, the user can issue `make install` which goes through all three steps sequentially using the make requirements order. Although the `config` step does not need to be done by the root user the install does, so doing this step as a user with root privileges is recommended.

```
# make config
Enter the AMSO1100 build path (/tmp/AMSO1100):
Base directory to install mpich (/usr/opt/ammasso):
Enter path to c compiler (/usr/bin/gcc):
Enter path to c++ compiler (/usr/bin/g++):
Enter path to fortran 77 compiler (/usr/bin/g77):
Build shared libraries (y or n)? [no]
Enter path to remote shell (/usr/bin/rsh):
Build mpich using a Fortran 90 compiler (y or n)? [no]

You are compiling mpich on a 64-bit operating system.  By default mpich is
only built natively.  If you wish to build a 32-bit version as well, please
say 'yes' below.  You only need a 32-bit version of mpich if you have
applications compiled as 32-bit binary only or you are unsure if your mpich
apps are 64-bit safe.  If you say 'yes' here, two mpich trees will be
installed into your /usr/opt/ammasso directory.  They will be called mpich-
1.2.5 and mpich-1.2.5-32.
Build 32-bit mpich (y or n)? [no]

# ls
.  ..  Config  Makefile  mpich-1.2.5  mpich_cset  scripts

# cat Config
CC='/usr/bin/gcc'
CPP='/usr/bin/g++'
FC='/usr/bin/g77'
STARCORE='/tmp/AMSO1100'
LDFLAGS=''
```

```
INSTALL_DIR='/usr/opt/ammasso/mpich-1.2.5'
RSHCOMMAND='/usr/bin/rsh'
ENABLE_SHAREDLIB='no'
F90=''
F90FLAGS=''
F90_LDFLAGS=''
PLATFORM='native'
CUSTOM_CFLAGS=''
BPROC='no'
#
```

3. Build MPICH.

```
# make
......
output from make, takes approximately five minutes
......
```

4. Install MPICH. This step needs to be done as the root user.

```
# make install
......
output of install
......

# ls /usr/opt/ammasso/
.  ..  bin  data  fw  lib  lib64  man  mpich-1.2.5  release  scripts
support
```

MPICH is now installed on the build system. If MPICH will be accessed via a distributed filesystem such as NFS export to the cluster nodes from the build system, the user can skip to the installing DAPL step below. However, if MPICH will be installed on each of the cluster nodes or will be exported from another system, first create a binary image file for installation on the cluster nodes.

5. Build a MPICH binary image for install on the cluster nodes. This step can also be done at a later time and after a reboot provided the AMSO1100 and AMSO_MPICH build directories are still present. The MPICH binary will install in the same base directory as specified in the `make config` step for MPICH. If a local install has already been done (via: '`make install`') the make binary needs to re-install to ensure a correct directory structure. Therefore the make binary will prompt to overwrite the existing installation. Say yes at this point if the MPICH installation directory (`/usr/opt/ammasso/mpich-1.2.5`) can be over-written and the make binary will re-install and make the binary images. If the MPICH installation directory has been modified it will be necessary to save that directory prior to doing the make binary and restore afterwards. To avoid the re-install, the installation and make binary can be done with the single make binary step.

```
# ls
.  ..  Build  Config  Installed  Makefile  mpich-1.2.5  mpich_cset  scripts

# make binary
......
output of make binary
......
It appears mpich has already been installed to /usr/opt/ammasso/mpich-1.2.5.
```

```
Overwrite existing mpich installation (y or n)? [no] y
......
output of install
......
Created binary self-extracting image --> /tmp/AMSO_MPICH/image-mpich-
1.2.5.bin
#

# ls
.   Build   image-mpich-1.2.5.bin  Makefile     scripts
..  Config  Installed              mpich_cset   mpich-1.2.5
```

`image-mpich-1.2.5.bin` is a shell script and binary image for installation on cluster nodes. Save this file to a safe location that can be used to distribute to the cluster nodes.

```
# cp image-mpich-1.2.5.bin /tmp
```

6.2.4 Install DAPL and Build Binary for Cluster Deployment

1. Use the `tar(1)` command to unarchive the DAPL package on the build system. The unzip/tar-extract for this example was done in `/tmp` but that is not a requirement. However placing the directory at the same location as the amso1100 build directory allows the configuration to automatically find it.

```
# cd /tmp
# ls
.               AMSO1100.tgz        .ICE-unix       YaST2-02308-zPffTu
..              AMSO_MPICH.tgz      .qt             3Ddiag.Yp3238
gconfd-root     sysconfig-update    AMSO1100        hps.test
.X11-unix       AMSO_DAPL.tgz

# tar zxf AMSO_DAPL.tgz

# ls
.               AMSO1100.tgz         hps.test          .X11-unix
..              AMSO_MPICH          .ICE-unix         YaST2-02308-zPffTu
3Ddiag.Yp3238   AMSO_MPICH.tgz      .qt
AMSO1100        gconfd-root         sysconfig-update
AMSO_DAPL.tgz   AMSO_DAPL

# cd AMSO_DAPL/

# ls
.  ..  Makefile  dapl-1.2  dapl_cset  etc  scripts
```

2. Build and install the AMSO_DAPL package. This step needs to be done as a user with root privileges. If a previous install has been done the make install will prompt whether it is okay to overwrite the potential installation at this location. The proper response is to answer `'y'`.

```
# cd /tmp/AMSO_DAPL/
# make install
......
output of build, make takes approximately 5 minutes
......
# ls /usr/opt/ammasso/
.   bin       data  lib64  mpich-1.2.5  scripts        support
..  dapl-1.2  fw    man    release      starcore_cset
```

3. Build an AMSO_DAPL binary package for install on the cluster nodes.

```
# ls
.  ..  Build  Config  Installed  Makefile  dapl-1.2  dapl_cset  etc  scripts
# make binary
Created binary self-extracting image --> /tmp/AMSO_DAPL/image-dapl-1.2.bin
# ls
.   Build   Installed  dapl-1.2   etc                scripts
..  Config  Makefile   dapl_cset  image-dapl-1.2.bin
```

`image-dapl-1.2.bin` is a shell script and binary image for installation on cluster nodes. Save this file to a safe location that can be used to distribute to the cluster nodes.

```
# cp image-dapl-1.2.bin /tmp
```

6.2.5 Clean up AMSO directories and files.

Before removing the AMSO_* directories and tar archive files, be sure that the binary image files `ammasso1100.bin`, `image-mpich-1.2.5.bin`, and `image-dapl-1.2.bin` have been saved in a separate location.

```
# cd /tmp

# ls
.               AMSO1100.tgz        hps.test   .X11-unix
..              AMSO_MPICH          .ICE-unix  YaST2-02308
3Ddiag.Yp3238   AMSO_MPICH.tgz      .qt
AMSO1100        gconfd-root         sysconfig-update
ammasso1100.bin  image-dapl-1.2.bin   image-mpich-1.2.5.bin

# rm -rf AMSO*
```

The build system can now be rebooted.

6.3 Steps on the cluster node systems

Now that the AMSO1100, MPICH, and DAPL packages have been built on the initial build system, only installations are required on the remaining cluster node systems (with the requirement that the distribution, patch level and processor match).

1. For each cluster node uninstall previous AMSO1100 package installations.

2. Copy or transfer the `ammasso1100.bin` script to the `/tmp` directory of the cluster node system. This can be done, for example, using `scp(1)` or `rcp(1)`, for example:

```
# scp app64-01:/tmp/ammasso1100.bin /tmp
```

3. On each cluster node, install the adapter software. The –d flag specifies which directory to use to hold the rnic_cfg file. This can be any directory on the system.

```
# cd /tmp/

# ls ammasso*
ammasso1100.bin
```

```
# /tmp/ammasso1100.bin -d /usr/opt/ammasso/data/app64-02
Ammasso 1100 binary installer created on ....

Using /usr/opt/ammasso/data/app64-02 as the configuration directory.
Installation complete.
Reboot the system to activate the AMSO1100 board and its software
# ls  /usr/opt/ammasso
.  ..  bin  data  fw  lib  lib64  man  release  scripts  starcore_cset
       support

# ls /usr/opt/ammasso/data
.  ..  app64-02  default   rnic_cfg.example

# ls /usr/opt/ammasso/data/app64-02/
.  ..  mode
```

4. Set the IP settings for the cluster node. Although the initial cluster node install involves a manual setting of the `rnic_cfg` file once it is set it will not be deleted by an `amso_uninstall` and will be reused by follow on `ammasso1100.bin` installs. This step needs to be done as a user with root privileges.

   Copy or transfer the `rnic_cfg` file from section 6.2.2 to the directory specified above. In this example, the directory is `/usr/opt/ammasso/data/app64-02`.

   Edit this file with the new address settings:

```
# pwd
/usr/opt/ammasso/data/app64-02

#tail rnic_cfg
function amso_rdma_0_0 {
   AMSO_IPADDR=10.40.32.50
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}
function amso_ccil_0_0 {
   AMSO_IPADDR=10.40.48.50
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}

# vi rnic_cfg

#tail rnic_cfg
function amso_rdma_0_0 {
   AMSO_IPADDR=10.40.32.51
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}
function amso_ccil_0_0 {
   AMSO_IPADDR=10.40.48.51
   AMSO_MASK=255.255.240.0
   AMSO_GW=
   AMSO_MTU=1500
}
```

5. On each cluster node, install the MPICH software. This step needs to be done as the root user.

   Copy or transfer the `image-mpich-1.2.5.bin` script to the /tmp directory of the cluster node system. This can be done, for example, using `scp(1)` or `rcp(1)`, for example:

```
# scp foo:/tmp/image-mpich-1.2.5.bin /tmp
# cd /tmp/

# ls image*
image-mpich-1.2.5.bin

# /tmp/image-mpich-1.2.5.bin

Ammasso Mpich binary installer created on ....

Mpich has been installed into /usr/opt/ammasso/mpich-1.2.5.

# ls  /usr/opt/ammasso
.  ..  bin  data  fw  lib  lib64  man  mpich-1.2.5  release  scripts
starcore_cset       support
```

6. On each cluster node, install the DAPL software. This step needs to be done as a user with root privileges.

   Copy or transfer the `image-dapl-1.2.bin` script to the /tmp directory of the cluster node system. This can be done, for example, using `scp(1)` or `rcp(1)`, for example:

```
# scp foo:/tmp/image-dapl-1.2.bin /tmp
# cd /tmp/

# ls image*dapl*
image-dapl-1.2.bin

# /tmp/image-dapl-1.2.bin

Ammasso Dapl binary installer created on ....

Dapl has been installed into /usr/opt/ammasso/dapl-1.2

# ls  /usr/opt/ammasso
.  ..  bin  dapl-1.2      data  fw  lib  lib64  man  mpich-1.2.5  release
scripts starcore_cset support
```

7. Clean up AMSO1100, MPICH, and DAPL cluster node tar files.

```
# cd /tmp

# ls *.bin
ammasso1100.bin  image-mpich-1.2.5.bin  image-dapl-1.2.bin

# rm ammasso1100.bin
# rm image-mpich-1.2.5.bin
# rm image-dapl-1.2.bin
```

AMSO1100, MPICH, and DAPL are now installed on the cluster node system. Repeat the above steps for each node in the cluster. Set RDMA and CCILNET addresses and reboot the systems.

6.4 Cluster Deployment

For deployment on a large number of clusters, a cluster administrator will want to script an installation procedure with:

1) an uninstall of previous Ammasso release software
2) an install of `ammasso1100.bin`, `image-mpich-1.2.5.bin`, and `image-dapl-1.2.bin`
3) place a new `rnic_cfg` for the system
4) reboot

The script will need to utilize `rsh(1)`, `ssh(1)` or perhaps `expect(1)` to access each cluster node.

7 Using the Ammasso 1100 with PXE Boot

PXE boot is a way to boot and load x86 based servers with a system image (or any other type of program) that is served from another computer on the network. It is especially useful when a unique operating system kernel image needs to be replicated across a number of servers. Another use is if an operating system kernel image needs to be loaded on remote servers that you do not have physical access to or which don't have an alternative boot device. PXE boot is available only for devices that implement the Intel Preboot eXecution Environment (PXE) specification. To determine if your server supports PXE network boot, see the hardware manufacturer's documentation for your motherboard.

7.1 Theory of Operation

The Ammasso 1100 supports PXE boot by default. There is nothing specific you need to do to enable the adapter[1]. When PXE boot support on the adapter is executed, the firmware sends out a DHCP request, and waits to be assigned it's own unique IP address as well as a filename and server that will provide the network boot program (NBP). It then downloads and runs the NBP. The firmware then downloads and runs the NPB.

For our testing, we used PXELINUX (http://syslinux.zytor.com) as our NBP. When PXELINUX runs, it downloads a configuration file from the TFTP server to find out the filenames of a linux kernel and associated ramdisk. PXELINUX then downloads them and finally boots the actual Linux kernel we wish to execute.

Once the kernel boots, the network driver used by the PXE boot firmware is no longer available. Therefore, if you use the network early in your kernel boot process, for instance to remotely mount you root filesystem via NFS, the user either needs a network driver statically built into the kernel image or provide a dynamically loaded one on the ramdisk. Since the AMSO1100's driver is not statically bound into the default kernel image, the driver must be included in the downloaded ramdisk and loaded into the running kernel during system start up. After the AMSO1100 driver is loaded and configured, the Ethernet can be used: `i.e.` you may remotely mount the root filesystem via NFS.

The following section provides some basic information about configuring the Ammasso adapter and software for use with PXE boot. The information provided is offered as a reference, as there may be multiple ways to accomplish these tasks, depending on your specific development environment.

7.2 Requirements

In order to use the Ammasso 1100 for PXE booting you will need the following set up. This list is provided for reference. Examples are provided in later sections which describe how to accomplish each of these.

[1] The Ammasso AMSO 1100 uses Etherboot (http://www.etherboot.org) to provide the PXE support. Etherboot generates an option ROM image that was factory flashed onto the network adapter.

1. A target machine that is PXE capable. The Ammasso 1100 card must be selected as the network boot device.
2. The Ammasso driver, `ccil.{o,ko}`, compiled against the kernel that will be PXE booted.
3. A ramdisk image that contains the Ammasso driver and utilities. The following files are required for the ramdisk to load. The Ammasso 1100 needs to have firmware loaded to it prior to loading the driver. The file locations are described below:

   Firmware:

   ```
   ${AMSO1100}/verbatim/fw/release/fw/boot_image
   ${AMSO1100}/verbatim/fw/release/fw/boot_image_start
   ```

   Firmware Loading Utility:

   ```
   ${AMSO1100}/verbatim/fw/boot
   ```

   Ammasso Kernel Module:

   2.4 Kernel:
   ```
   ${AMSO1100}/software/host/linux/sys/devccil/obj_`uname -
r`_release/ccil.o
   ```

   2.6 Kernel:
   ```
   ${AMSO1100}/software/host/linux/sys/devccil/obj_`uname -
r`_release/ccil.ko
   ```

   The firmware and firmware loading utility can be copied anywhere into the ramdisk. We recommend putting them into "`/ccore/`.".

   The Ammasso kernel module needs to be copied into the network modules location of the ramdisk file system.

4. The init scripts of the ramdisk file system, commonly called the rc init scripts, will need to be modified to get the Ammasso 1100 to load. As noted before, the firmware must be loaded prior to loading the kernel module.

   To load the firmware, we use the boot program. "`boot`" requires two firmware files as arguments. The command to load the firmware is:

   ```
   /<path>/boot /<path>/boot_image 0x`cat \
   /<path>/boot_image_start`
   ```

   Once the firmware has been loaded you can insmod the ccil module:

   ```
   insmod /<path>/ccil.{o,ko}
   ```

5. A system configured to be a PXE boot server. This system must be running DHCP (dynamic host configuration protocol), TFTP (trivial file transfer protocol), and will need to have a remote file service available (such as NFS - network file system).

7.3 BIOS Settings

When the motherboard is initialized, it locates all option ROMs and executes BIOS code in each ROM if it is enabled in the BIOS. If PXE is enabled in your motherboard's BIOS, the Ammasso option ROM will begin to execute the PXE support. However, each user must enable PXE boot in the boot menu of your motherboard's BIOS. See your motherboard documentation on how to do this. On some motherboards, the users must also adjust the boot device priority list so that network boot is attempted before booting from other devices such as the hard disk, CD-ROM, floppy, *etc.*.

The Ammasso 1100 adapter is listed in the BIOS boot list as:

```
amm_ccore.zrom 5.3.8 (GPL) etherboot.org
```

Once PXE Boot is enabled, you can create a single system image in one location and have your remote machine load that image. At boot time, the server will present the following:

```
Boot from (N)etwork or (Q)uit?
```

The default, '`N`', is used if nothing on the keyboard is depressed, causing the system to perform a network boot. If '`Q`' is depressed, the PXE boot code will be disabled and the system will boot normally.

7.4 Building the RAMDISK

The ramdisk contains programs that are used to help in the start up process. When booting from hardware disk, these utility programs are built and installed in the traditional manner. For PXE style environments, extremely small statically bound versions of these have been created to help keep the size of the ramdisk as small as possible, so as to not inversely affect load time over the network.

To provide the executables needed in the ramdisk, the version developed by BusyBox (`http://www.busybox.net`) can be used for most applications. Due to some integration issues, we use the standard Linux `insmod` from `modutils` or `module-init-tools` rather than those provided by BusyBox. These versions can be downloaded from the following web sites:

`http://www.kernel.org/pub/linux/utils/kernel/modutils/v2.4/`
[for 2.4 kernels]

`http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/`
[for 2.6 kernels]

`http://busybox.net/downloads/`

To simplify building, all applications are shown to be linked statically.

For the next few steps we assume that the shell variables, BUSYBOX and MODUTILS, have been set to the path of the sources for their namesakes. The variable INITRD should be set to a temporary directory where the ramdisk will be built.

7.4.1 Configuring and Building BusyBox Applications

BusyBox is configured like the Linux kernel, using:

```
# make config
```

or:

```
# make menuconfig
```

Once the `.config` file has been generated in the BusyBox source directory, build the BusyBox applications with:

```
# cd ${BUSYBOX}/
# make dep
# make
# make install
```

This places the BusyBox applications into the `${BUSYBOX}/_install/` directory

7.4.2 Building modutils or module-init-tools

You only need the statically linked `insmod(8)`. First, configure `modutils` for 2.4 kernels or `module-init-tools` for 2.6 kernels:

```
# cd ${MODUTILS}/
# ./configure --disable-insmod-static \
--enable-combined --enable-strip --disable-combined
```

Next, build it:

```
# make
```

At the point, the `make(1)` utility may generate errors. However, we've found those errors to be in sections of the BusyBox utilities that are not needed for the ramdisk. After `make(1)` returns perform the following steps:

```
# cd insmod
# make insmod.static
# strip insmod.static
```

7.4.3 Populating the Ramdisk

Create an initrd directory structure to be used for the ramdisk. For this example, the initrd directory structure is shown below:

```
# ls ${INITRD}
bin  ccore  dev  etc  linuxrc  mnt  modules  proc  sbin  share  var
```

Copy the BusyBox files:

```
# cp -a ${BUSYBOX}/_install/bin/* ${INITRD}/bin
# cp -a ${BUSYBOX}/_install/sbin/* ${INITRD}/sbin
```

Copy `insmod` from `modutils`:

```
# cp ${MODUTILS}/insmod/insmod.static ${INITRD}/bin/insmod
```

Copy the firmware and loading tools:

```
# cp ${AMSO1100}/verbatim/fw/boot ${INITRD}/ccore/ccos.boot
# cp ${AMSO1100}/verbatim/release/fw/boot_image ${INITRD}/ccore/ccos.bin

# cp ${AMSO1100}/verbatim/release/fw/boot_image_start \
      ${INITRD}/ccore/ccos.bin_start
```

Copy the `AMSO1100` host device driver. This driver module must be built against the kernel that will be PXE booted.

```
# cp ${AMSO1100}/software/host/linux/sys/devccil/obj_\
      `uname -r`_release/ccil.o ${INITRD}/modules/net/
```

Depending on your kernel configuration, you may need to copy additional drivers to the ramdisk. These drivers can be found in `/lib/modules/`uname -r`/kernel/`

With most RedHat installations we needed drivers to support NFS. The modules we needed were `sunrpc.{o,ko}`, `lockd.{o,ko}`, and `nfs.{o,ko}`. They should be installed in `${INITRD}/modules/net/.`

With most SuSE installations we needed drivers for Raw Socket support. The module we needed was `af_packet.{o,ko}`. This module is installed in `${INITRD}/modules/net/.`

Edit `${INITRD}/linuxrc` accordingly (see below for some detail).

Example 1 [for RHEL 3, 2.4.21-15.ELsmp]:

```
# load the drivers
insmod /modules/net/sunrpc.o
insmod /modules/net/lockd.o
insmod /modules/net/nfs.o
insmod /modules/net/ccil.o
```

Example 2 [for SuSE 9, 2.4.21-102-smp]:

```
# load the drivers
insmod /modules/net/af_packet.o
```

```
insmod /modules/net/ccil.o
```

Example 3 [FC 3, 2.6.9-1.667smp]:

```
# load the drivers
insmod /modules/net/sunrpc.ko
insmod /modules/net/lockd.ko
insmod /modules/net/nfs.ko
insmod /modules/net/ccil.ko
```

The following is the `linuxrc` file used in this example:

```
#!/bin/bash

export PATH=/bin:/sbin:

mount -t proc none /proc
mount -t devpts none /dev/pts

# Parse the kernel command line
cmdline="$(cat /proc/cmdline 2>/dev/null)"
for i in $cmdline; do case "$i" in
ip=*|IP=*|nm=*|NM=*|gw=*|GW=*|nfsdir=*|NFSDIR=*) eval $i;; esac; done
[ -n "$ip" ] && IP="$ip"
[ -n "$nm" ] && NM="$nm"
[ -n "$gw" ] && GW="$gw"
[ -n "$nfsdir" ] && NFSDIR="$nfsdir"

# Load the NFS drivers
insmod /modules/net/sunrpc.ko
insmod /modules/net/lockd.ko
insmod /modules/net/nfs.ko

# Load the RAW socket support driver
#insmod /modules/net/af_packet.ko

# Load the Ammasso 1100 firmware
echo /ccore/ccos.boot /ccore/ccos.bin 0x`cat /ccore/ccos.bin_start`
/ccore/ccos.boot /ccore/ccos.bin 0x`cat /ccore/ccos.bin_start`

# Load the ccilnet driver
insmod /modules/net/ccil.ko

# Define the interface name
export IFACE=ccilnet0

ifconfig lo 127.0.0.1 up
ifconfig
lsmod

# Make 100 DHCP requests
for i in 1 2 3 4 5 6 7 8 9 10; do
    for j in 1 2 3 4 5 6 7 8 9 10; do
        echo udhcpc -q -n -i $IFACE
        udhcpc -q -n -i $IFACE && DHCP="true"
        [ -n "$DHCP" ] && break
        echo "No answer from network"
    done
done

# Break to a shell if we don't get a DHCP response
[ -z "$DHCP" ] && /bin/ash

# Display the interface configuration
ifconfig $IFACE
```

```
# Display the routing table
route -n

# Mount what will become the root filesystem
mount -t nfs -o nolock,rsize=8192,wsize=8192,hard,intr ${NFSDIR} /mnt

# We don't need these anymore, and they will get remounted later, anyway
umount /dev/pts
umount /proc

# Switch the mount points - the NFS mounted FS will become /
cd /mnt
pivot_root . initrd

# Mount all
mount -a
echo 256 > /proc/sys/kernel/real-root-dev

# Return, and init on the newly mounted filesystem will continue
# the boot process.
```

7.4.4 Building the Ramdisk Image

First, build a blank ramdisk image. In this example, we use 16MB. If additional software is required, the ramdisk image may need to be larger.

```
# dd if=/dev/zero of=${PXEROOT}/initrd.img bs=1024K count=16
```

Next, format the ramdisk:

```
# mke2fs -F -m0 -b 1024 ${PXEROOT}/initrd.img
```

Then, mount it and copy the data to it:

```
# mkdir /mnt/initrd/
# mount -o loop ${PXEROOT}/initrd.img /mnt/initrd/
# tar cvf - ${INITRD} |( cd /mnt/initrd/; tar xvfp - )
```

Lastly, unmount it and compress it:

```
# umount /mnt/initrd/
# gzip -9 ${PXEROOT}/initrd.img
```

7.5 Installing and Configuring a DHCP Server

A DHCP (Dynamic Host Configuration Protocol) server is required on your network. There are HOWTOs covering installation and configuration at the following web site (`dhcpd` also comes with most distributions).

http://www.tldp.org/HOWTO/DHCP/index.html

We recommend that you ensure that your external firewall blocks port 67 and 68 for both UDP and TCP since DHCP has no security.

There needs to be one entry for each MAC address in the "group" section of the file. A sample `dhcpd.conf` configuration file is shown here:

```
authoritative;
ddns-update-style interim;
option domain-name "ammasso.com";

shared-network lab {

# cn0 subnet

subnet 10.40.48.0 netmask 255.255.240.0 {
    option routers 10.40.48.1;
    option broadcast-address 10.40.63.255;
    option domain-name-servers 10.40.48.1;
    option subnet-mask 255.255.240.0;

    use-host-decl-names true;

     host sqa-14-cn0 {
         fixed-address 10.40.48.171;
         hardware ethernet 00:0D:B2:00:06:72;
         filename "/pxelinux.0";
     }
   }
}
```

7.6 Installing and Configuring a TFTP Server

You will need a TFTP (Trivial File Transfer Protocol) server that supports the "`tsize`" option. The one we used is Peter Anvin's tftp server:

http://www.kernel.org/pub/software/network/tftp/tftp-hpa-0.36.tar.gz

We recommend that you ensure that your external firewall blocks port 69 for both UDP and TCP since tftp has no security.

We made a world readable directory `/tftpboot` for the TFTP daemon to serve files from.

To enable it on RedHat, which uses `xinetd`, we created `/etc/xinetd.d/tftpd`, with the following contents:

```
service tftp
       {
              disable = no
              socket_type = dgram
              protocol = udp
              wait = yes
              user = root
              server = /usr/local/sbin/in.tftpd
              server_args = -s /tftpboot
              port = 69
       }
```

Restart `xinetd` after adding the daemon.

7.7 Building PXELINUX

In this example, we use PXE to boot Linux with PXELINUX from the `syslinux` package. This can be obtained from the following website:

http://www.kernel.org/pub/linux/utils/boot/syslinux/

To build `PXELINUX`, you will need `nasm(1)`, an x86 assembler; which can be obtain from the following website:

http://sourceforge.net/projects/nasm

You only need to build `pxelinux.0`. To build it:

```
# make pxelinux.0
```

Copy `pxelinux.0` to your `/tftpboot` directory on your TFTP server. Next, copy your ramdisk to `/tftpboot`:

```
# cp ${INITRD}/initrd.img.gz /tftpboot/
```

Copy your kernel to `/tftpboot/` and copy the filesystem you want to use to `/tftpboot/`.

Create a config file named "`default`" for PXE. It goes in the directory `/tftpboot/pxelinux.cfg/`

Example:

```
KERNEL vmlinuz-2.4.20-6smp
APPEND nfsdir=10.40.0.1:/NFSroot/client1 init=/sbin/init \
initrd=initrd.img.gz
TIMEOUT 10
```

`KERNEL` specifies the kernel image
`APPEND` needs the IP address of the NFS server, the location of the filesystem, and the location of the ramdisk we created for AMSO1100.

More info on PXELINUX at http://syslinux.zytor.com/pxe.php

7.8 Diskless Linux Boot via PXE

There are several HOWTOs that explain booting a diskless node via PXE. Eg:
http://www.intra2net.com/opensource/diskless-howto/howto.html

7.8.1 Configure a Root File System

The idea behind a diskless node is to use the distribution that has been installed on a remote server via NFS. We used a two step process to create the root file system for a client on the server.

1. Copy all files from the distribution installed on a disk into a directory on the server. `NFSROOT=/NFSroot/clients/OS/client1`. The directory `/NFSroot/clients/OS/client1` is the distribution root directory. In addition, create a directory `/NFSroot/clients/OS/client1/initrd` since we make use of this in the ramdisk with pivot_root.

2. Prepare the copied distribution directory for network booting.

    a. Replace the entry for `/` in `${NFSROOT}/etc/fstab` with
    ```
    none    /       tmpfs       defaults        0 0
    ```
    This will prevent an `fsck.ext2`

    b. Ensure that all NFS mounts in `${NFSROOT}/etc/fstab` have the `nolock` option turned on.

    c. Disable networking (we set this up in the ramdisk over AMSO1100)
    ```
    /usr/sbin/chroot ${NFSROOT} /sbin/chkconfig \
    --del network
    ```

    d. Create the following symbolic links since `/etc` will be read-only
    ```
    ln -sf /proc/mounts ${NFSROOT}/etc/mtab
    ln -sf /var/resolv.conf \
    ${NFSROOT}/etc/resolv.conf
    ```

More info on creating NFS root filesystems can be obtained at:
http://www.tldp.org/HOWTO/NFS-Root-Client-mini-HOWTO/

7.8.2 Configure the NFS Server

The NFS server should export the distribution directory. Add a line like this to your `/etc/exports` on the NFS server:
```
/NFSroot/clients/OS/client1 \
10.40.48.0/255.255.0.0(ro,no_root_squash)
```
Run `/usr/sbin/exportfs -ra` after you have added the line and restart the NFS daemon. This directory should not be mounted in read-write mode since you need to set `no_root_squash`.

More info on configuring an NFS server can be found at the following web site:
http://www.tldp.org/HOWTO/NFS-HOWTO/

7.9 Updating the Ammasso 1100 Option ROM Image

The Ammasso 1100 network adapter was factory flashed with an option ROM image. If you are updating from a previous release, this image will need to be updated (see `HOWTO_UPDATE.txt` for more specifics on updating the software and hardware). Ammasso provides utilities which allow this image to be updated by the user. The `ccflash2` utility can be used to update images on the Ammasso 1100 network adapter. In order to update the option ROM image, the `-pxe` flag should be used. The following is an example of updating the option ROM:

```
ccflash2 -t pxe 0 ${AMSO1100}/verbatim/fw/amm_ccore.zrom
```

Appendix A: Support

Obtaining Additional Information

Additional information, the latest software revisions, and documentation is always available at the Ammasso Customer Support website located at:

http://www.ammasso.com/support

Contacting Ammasso Customer Support

If you have a question concerning your Ammasso 1100 High Performance Ethernet Adapter or driver software, refer to the technical documentation supplied with your Ammasso 1100 Adapter. If you need further assistance, contact your Ammasso supplier. Ammasso Customer Support can be reached using the following contact information:

E-mail: support@ammasso.com
Telephone: 617-532-8110
Fax: 617-532-8199
http://www.ammasso.com/
Post mail: Ammasso, Inc.
345 Summer St.
Boston, MA 02210

Returning a Product to Ammasso

Ammasso requires its customers to obtain a RMA number from Support to return a product. Once received, the customer is expected to properly package the product to ensure safe return to the designated facility. Ammasso, Inc will not be responsible for any damage to the adapter incurred during shipment, or due to inadequate packaging. Be sure to use the original package when returning products.

Appendix B: Warranty

Ammasso High Performance Server Adapter

LIMITED LIFETIME HARDWARE WARRANTY

Ammasso warrants to the original owner that its adapter product will be free from defects in material and workmanship. This warranty does not cover the adapter product if it is damaged in the process of being installed or improperly used.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF NONINFRINGEMENT OF INTELLECTUAL PROPERTY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SPECIFICATION, OR SAMPLE.

This warranty does not cover replacement of adapter products damaged by abuse, accident, misuse, neglect, alteration, repair, disaster, improper installation, or improper testing. If the adapter product is found to be defective, Ammasso, at its option, will replace or repair the hardware product at no charge except as set forth below, or refund your purchase price provided that you deliver the adapter product along with a Return Material Authorization (RMA) number along with proof of purchase, either to the reseller from whom you purchased it with an explanation of any deficiency. If you ship the adapter product, you must assume the risk of damage or loss in transit. You must use the original container (or the equivalent) and pay the shipping charge.

Ammasso may replace or repair the adapter product with either new or reconditioned parts, and any adapter product, or part thereof replaced by Ammasso becomes Ammasso property. Repaired or replaced adapter products will be returned to you at the same revision level as received or higher, at Ammasso's option. Ammasso reserves the right to replace discontinued adapter products with an equivalent current generation adapter product.

LIMITATION OF LIABILITY AND REMEDIES

AMMASSO'S SOLE LIABILITY IN CONNECTION WITH THE SALE, INSTALLATION AND USE OF THE ADAPTER SHALL BE LIMITED TO DIRECT, OBJECTIVELY MEASURABLE DAMAGES. IN NO EVENT SHALL AMMASSO HAVE ANY LIABILITY FOR ANY INDIRECT CONSEQUENTIAL, INCIDENTAL, OR SPECIAL DAMAGES, REPROCUREMENT COSTS, LOSS OF USE, BUSINESS INTERRUPTIONS, LOSS OF GOODWILL, OR LOSS OF PROFITS, WHETHER ANY SUCH DAMAGES ARISE OUT OF CONTRACT NEGLIGENCE, TORT, OR UNDER ANY WARRANTY, IRRESPECTIVE OF WHETHER AMMASSO HAS ADVANCE NOTICE OF THE POSSIBILITY OF ANY SUCH DAMAGES. NOTWITHSTANDING THE FOREGOING,

AMMASSO'S TOTAL LIABILITY FOR ALL CLAIMS UNDER THIS AGREEMENT SHALL NOT EXCEED THE PRICE PAID FOR THE PRODUCT. THESE LIMITATIONS ON POTENTIAL LIABILITIES WERE AN ESSENTIAL ELEMENT IN SETTING THE PRODUCT PRICE. AMMASSO NEITHER ASSUMES NOR AUTHORIZES ANYONE TO ASSUME FOR IT ANY OTHER LIABILITIES.

Critical Control Applications: Ammasso specifically disclaims liability for use of the adapter product in critical control applications (including, for example only, safety or health care control systems, nuclear energy control systems, security or defense systems, or air or ground traffic control systems) by Licensee or Sublicensees, and such use is entirely at the user's risk. Licensee agrees to defend, indemnify, and hold Ammasso harmless from and against any and all claims arising out of use of the adapter product in such applications by Licensee or Sublicensees.

Software: Software provided with the adapter product is not covered under the hardware warranty described above. If the Software has been delivered by Ammasso on physical media, Ammasso warrants the media to be free from material physical defects for a period of ninety (90) days after delivery by Ammasso. If such a defect is found, return the media to Ammasso for replacement or alternate delivery of the Software as Ammasso may select. Software licenses for the Ammasso 1100 Ethernet Adapter can be found at www.ammasso.com, or with the Ammasso software.